DEEP LEARNING TECHNIQUES FOR IMAGE SEGMENTATION OF WHOLE BODY MRI

by

Alex Enrico Bocchieri

A thesis submitted to Johns Hopkins University in conformity with the requirements for the degree of Master of Science in Engineering

Baltimore, Maryland

May 2020

Abstract

Recently, deep learning methods have achieved state of the art results across many fields of research. Deep learning has become prevalent in medical image analysis and plays an integral part of contemporary healthcare. In particular, contemporary image segmentation techniques make extensive use of deep artificial neural networks.

This thesis presents two deep learning architectures for segmenting 3D multiparametric MRIs of patients with limb girdle muscular dystrophy. Muscular dystrophy is a genetic disease that causes muscle to be replaced by fatty tissue over time. Segmenting images of patients with muscular dystrophy has not been done prior to this work, and no labeled datasets of muscular dystrophy images are openly available. Image segmentations provide quantitative metrics, such as the fraction of muscle voxels out of all non-background voxels, that can help clinical researchers track the progression of the disease in a patient.

The fact that a labeled dataset does not exist poses a challenge. A relatively small, manually labeled dataset of tissue signatures is created to train convolutional neural networks and stacked sparse autoencoders. The success of

these deep learning models relies on the high contrast provided by multiparametric MRIs for distinguishing voxels of different classes. The models achieve excellent segmentation accuracy in slices of the thighs.

In addition, a deep reinforcement learning framework for locating anatomical landmarks is developed as a step toward improving the segmentation models' performance. A single agent model and a multi-agent model for landmark localization are demonstrated to successfully operate in multiple anatomical regions in a diverse set of imaging modalities and image acquisition parameters. Automatic landmark localization can be used for improving segmentations in low contrast areas as well as for automatic image registration algorithms. This thesis establishes a baseline for a universal deep reinforcement learning framework for such applications.

Advisors: Dr. Michael Jacobs and Dr. Vladimir Braverman

Acknowledgments

I am extremely grateful to Dr. Michael Jacobs for his continuous guidance and encouragement throughout this project. He immediately made me feel part of the team and transmitted to me his enthusiasm in research. It was a true pleasure to work under his supervision. I am thankful to him for providing a base from which the direction of the present work branched off and for always being available for suggestions and discussion. He also generously gave me access to his workstation and DGX machine, which afforded me the computational resources needed to complete this work.

I am deeply indebted to Dr. Vladimir Braverman for invaluable discussion of my work and for inviting me to participate in his group meetings. Attending his lectures and group meetings greatly increased my knowledge of theoretical fields connected to my thesis. They also provided me with insight into how collegial scientific discussions are conducted.

I wish to express my deepest gratitude to Dr. Vishwa Parekh for his constant help, guidance, and friendly support throughout this endeavor. I very much enjoyed planning experiments and discussing results with him. Besides the scientific advice, I could rely on his generous help when I encountered hurdles that would otherwise bog me down.

I wish to thank Dr. Shivani Ahlawat for providing ground truth segmentations and further discussion from a practicing radiologist's perspective.

I am also grateful to numerous faculty and friends who made it a pleasure for me to live, study, and work at Johns Hopkins University over the past several years.

Finally, I am truly grateful to my parents for their unconditional love, constant encouragement, and financial support.

List of Publications

Major contributions of this thesis have been published (or are under review) in the manuscripts listed below.

1. V. S. Parekh, J. Laterra, C. Bettegowda, **A. E. Bocchieri**, J. J. Pillai, M. A. Jacobs. "Multiparametric Deep Learning and Radiomics for Tumor Grading and Treatment Response Assessment of Brain Cancer: Preliminary Results." arXiv preprint arXiv:1906.04049 (2019).

2. **A.E. Bocchieri**, V. S. Parekh, K. R. Wagner, S. Ahlawat, V. Braverman, D. G. Leung, M. A. Jacobs. "Multiparametric Deep Learning Tissue Signatures for Muscular Dystrophy: Preliminary Results." arXiv preprint arXiv:1908.00175 (2019).

3. V. S. Parekh*, **A. E. Bocchieri***, V. Braverman, M. A. Jacobs. "Multitask radiological modality invariant landmark localization using deep reinforcement learning." Submitted to MIDL 2020. (* denotes equal contribution)

4. D. G. Leung, **A. E. Bocchieri**, S. Ahlawat, M. A. Jacobs, V. S. Parekh, V. Braverman, K. Summerton, J. Mansour, G. Bibat, C. Morris, S. Marraffino,

K. R. Wagner. "Longitudinal functional and imaging outcome measures in FKRP limb-girdle muscular dystrophy." Submitted to BMC Neurology.

5. **A. E. Bocchieri**, V. S. Parekh, K. R. Wagner, S. Ahlawat, V. Braverman, D. G. Leung, M. A. Jacobs. "Multiparametric Deep Learning Tissue Signatures for Segmentation of Muscular Dystrophy." In preparation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Image Segmentation using Deep Learning

Medical image analysis has been an active research area for decades. Early approaches were predominantly based on traditional image processing techniques, such as filtering, thresholding, histogram equalization, etc. Around the 1990s, machine learning approaches began to emerge in various image analysis methods. Active contours (snakes) [1] for image segmentation were introduced in the late 1980s. Fundamental improvements to active contours in their efficiency and stability soon followed [2, 3, 4]. Atlas methods [5, 6, 7] use manually labeled images and fit them to unlabeled images for segmentation. Graph cuts are also commonly used for image segmentation [8, 9, 10, 11]. Traditional methods often used models that operate on handcrafted features from the input data. Developing discriminative handcrafted features can be a difficult endeavor on its own. In contrast, contemporary deep learning models, such as artificial neural networks, learn meaningful features on their own without the need of features to be handcrafted by a human.

Research in artificial neural networks first took place in the 1960s and

was centered on multi-layer perceptrons [12]. Convolutional neural networks (CNN) were first introduced in 1980 [13] and had a brief revival in the 1990s when a CNN, named LeNet, was used for classifying handwritten digits [14]. Training techniques and computing systems at the time were still inadequate for efficient use of neural networks. Today's widespread use of neural networks started in 2012 when a CNN, named AlexNet, obtained the best performance in the ImageNet challenge of classifying images of 1000 different classes [15].

1.1 Medical Image Segmentation using Deep Learning

The objective in medical image segmentation is to assign individual voxels to a certain class. In addition to the visual enhancement of regions of interest (e.g. organ, anatomical object, spiculation, etc.), segmentations provide quantitative metrics that are used by computer-aided diagnosis systems and doctors/researchers. For example, the segmentation of a cardiac MRI into background, left ventricle, right ventricle, and myocardium during diastole and systole provides volumetric measures that a diagnosing algorithm (or doctor) can use to diagnose possible heart diseases. Similarly, segmentations can provide shape metrics of spiculations for diagnosing breast cancer.

While many segmentation methods exist, contemporary methods predominantly use deep learning and artificial neural networks. The CNN is among the most popular deep learning architectures. Common CNN architectures include patch-based CNNs and fully convolutional networks (FCNs). The

stacked sparse autoencoder (SSAE) is another architecture that can be used for segmentation tasks. The following sections consider these architectures in the view of segmentation tasks.

1.2 Neural Network Background

A CNN is typically composed of convolutional, fully connected, pooling, and activation layers. Batch normalization [16] and dropout [17] layers are also commonly used to stabilize training and reduce overfitting respectively. The conventional deep CNN architecture has a sequence of convolution and pooling layers followed by final fully connected layers and a softmax classification layer. An activation layer follows each convolutional and fully connected layer. When no fully connected layer is used, the CNN is a fully convolutional network (FCN).

1.2.1 Convolutional Layer

A convolutional layer takes an image or feature map as input and produces a feature map as output. I describe the case of 2D convolution, and the 3D case follows by including an extra dimension (depth). The input X is of size $C_{in} \times H \times W$, where C_{in} is the number of input channels, and $H \times W$ is the height and width of the 2D image. Each filter in the layer produces one output channel. Therefore, a layer with k filters produces $C_{out} = k$ channels in the output feature map. If $C_{in} = 1$, then one filter is simply $M \times N$ in size, and the filter's output channel is the convolution of the input and the filter. Typically, $M = N$. If $C_{in} > 1$, then one filter is $C_{in} \times M \times N$ in size,

so that there is one $M \times N$ kernel for each input channel. For one filter, an $M \times N$ kernel is convolved with one $H \times W$ input channel for each of the C_{in} channels. Each of the C_{in} convolutions' outputs are summed element-wise, which yields the filter's output feature channel. Each filter in the layer produces one output feature channel in this manner. The filters' weights are learned during training. Therefore, there are $k \times C_{in} \times M \times N$ trainable parameters in one convolutional layer. All kernels have different (trainable) weights, including the different kernels within one filter. A learnable bias b for each filter can also be included to reduce overfitting. The ith output feature channel is computed as $W_i * X + b_i$ for $i = 1...C_{out}$, where W_i is the ith filter's weights, and b_i is the bias.

The entire kernel must lie in the input image when performing the convolution operation. Therefore, the output feature map will be smaller than the input in the $H \times W$ dimensions. If desired, the input can be zero-padded such that the output feature map has the same size as the input.

1.2.2 Fully Connected Layer

A fully connected layer generates a vector $y \in \mathbb{R}^N$, where each y_i is a linear combination of the previous layer's output elements $X \in \mathbb{R}^M$. $y = WX + b$, where $W \in \mathbb{R}^{N \times M}$ is the weight matrix and $b \in \mathbb{R}^N$ is the bias vector. W and b are learned during training.

1.2.3 Pooling Layer

A pooling layer is used to downsample an input image or feature map X. A pooling layer has a kernel of size $M \times N$ that iterates over X in a sliding window fashion, like in convolution. Typically, $M = N$. For each $M \times N$ region in X covered by the kernel, the pooling operation outputs one value (pixel). A max-pooling operation outputs a region's maximum value. An average-pooling operation outputs a region's average value. The kernel's stride is usually set so that regions do not overlap. For example, a 2×2 kernel with stride 2 has no overlapping regions and will downsample the input by a factor of 2.

A convolutional layer is sensitive to its input features' locations. Therefore, downsampling makes CNNs more robust to small rotations, translations, etc. in input images. Deep convolutional layers whose inputs have lower resolution learn more abstract features. Too much downsampling, however, will cause features to be lost entirely.

1.2.4 Activation Layer

An activation layer applies a nonlinearity to its input. Common activation functions include ReLU [18], sigmoid, and tanh. ReLU is defined as $f(x) = \max(0, x)$, and sigmoid is defined as $f(x) = (1 + e^{-x})^{-1}$. Backpropagation uses the network's gradients for updating weights, which is why properties of activations functions, such as their slopes, are considered when selecting which activation to use. ReLU is currently the most commonly used nonlinearity because its slope is 1 for $x > 0$, and it helps reduce effects of the

vanishing gradient problem [19]. Activations are what allow neural networks to learn complex nonlinear mappings between input features and output labels.

The final layer in the neural network for classification is an activation layer that prepares a prediction $\hat{y}$ to be input into the loss function, given a target y. In the case of multi-class classification, $\hat{y}$ and y are vectors. Certain loss functions require $\hat{y}$ to be a probability vector, so that $\hat{y}_i \in (0,1)$ and $\sum_{i=1}^{N} y_i = 1$ for N-class classification. For these loss functions [equations 1, 4, 5], the softmax activation is commonly used. The softmax function outputs the probability that y_i is the input image's class: $\Pr(y_i) = f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$ where x is the previous layer's output of length N. The sigmoid function also outputs values between 0 and 1, but not as probabilities (values do not sum to 1). Functions like sigmoid and ReLU do not output probabilities and are usually not used in the final classification layer.

1.2.5 Loss Functions

A neural network's weights are updated during training using gradient descent and backpropagation. In gradient descent, weights w (model parameters) are iteratively updated to minimize a loss function L. A single step in gradient descent updates weights $w_{t+1} = w_t - \gamma \nabla L(w_t(x))$, where $\gamma \in (0,1)$ is the learning rate and t is the time step. γ is often 0.01 or less. During training, loss L is computed in the final classification layer during a network's forward pass using output $\hat{y} = w(x)$ and ground truth y (training label) for a given input x (training sample). Table 1.1 summarizes commonly used loss functions. Many

class-weighted variations and other loss functions exist as well.

Regularization terms, such as L_1 and L_2, can be added to a loss function to increase bias and reduce overfitting. L_1 regularization drives model weights w to a more sparse solution by adding the term $\lambda \sum_j |w_j|$ to the loss function. L_2 regularization reduces the magnitude of the weights by adding the term $\lambda \sum_j w_j^2$ to the loss function. λ is the regularization coefficient and w_j is the jth weight in the model.

Loss Function	Equation
[1] Cross Entropy	$-\sum_{i=1}^{N} y_i \log \hat{y}_i$
[2] Huber	$\begin{cases} 0.5(y-\hat{y})^2 & \text{if } \lvert y-\hat{y}\rvert < 1 \\ \lvert y-\hat{y}\rvert - 0.5 & \text{otherwise} \end{cases}$
[3] Hinge	$\max(0, 1-y\hat{y})$
[4] KL Divergence	$\sum_{i=1}^{N} y_i \log(y_i/\hat{y}_i)$
[5] Negative log likelihood	$-\log \sum_{i=1}^{N} y_i \hat{y}_i$
[6] Dice	$1 - \frac{2\sum_i^N p_i g_i}{\sum_i^N {p_i}^2 + \sum_i^N {g_i}^2}$
[7] Mean Abs. Error	$\frac{1}{N}\sum_{i=1}^{N} \lvert y_i - \hat{y}_i\rvert$
[8] Mean Sq. Error	$\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

Table 1.1: Summary of common loss functions. Prediction $\hat{y}$ and target y may be length N vectors. $y \in [0,1]$, but is typically one-hot for classification tasks. $\hat{y} \in (0,1)$. For hinge loss, $y \in \pm 1$, $\hat{y} \in (-\infty, \infty)$. In Dice loss for binary image segmentation, p_i is the prediction and g_i is the ground truth for the ith pixel. $p_i, g_i \in \{0,1\}$. For MAE and MSE, y, $\hat{y} \in (-\infty, \infty)$.

1.3 Patch-Based CNN

The patch-based CNN operates on a square patch from a local region of an image and classifies the patch's center pixel. One forward pass in the CNN classifies one pixel. The CNN individually classifies each pixel in a sliding window fashion to obtain a segmentation for the entire image. A patch-based CNN is typically composed of convolutional, pooling, activation, fully connected (linear) layers, and a final softmax layer for classification. When classifying pixels that are close to the image's boundary, the input patch may extend outside the image boundary. In this case, the image can be zero-padded, or pixels can be mirrored so that input patch to the CNN is still valid.

Ciresan et al. [20] used a patch-based CNN to perform binary segmentation (membrane versus non-membrane) of 3D electron microscopy images of the brain. They trained various models with square patch sizes of 65 and 95 and segmented a $512 \times 512 \times 30$ volume on a slice-by-slice basis. Their method outperformed all other competing techniques in terms of the three metrics, rand error, warping error, and pixel error, in the ISBI 2012 EM Segmentation Challenge. A significant difference in their architecture compared to early CNNs is that they used max-pooling layers instead of subsampling layers. A max-pooling layer maps small regions (usually non-overlapping) of an image to a single pixel whose value is the region's maximum value. Subsampling and average-pooling both take the region's average value as the single pixel's value. Early CNNs [13, 14, 21] commonly used subsampling between convolutional layers. Scherer et al. [22] empirically determined that max-pooling operations

significantly outperform subsampling operations. Papers such as [23, 24] used max-pooling, as do most contemporary CNNs. Pooling operations provide robustness to translations in input images, help decrease over-fitting, and reduce dimensionality to decrease computational costs.

Hou et al. [25] classified cancer subtypes in very high resolution whole slide tissue images using a patch-based CNN. Due to the images' very high resolution, a conventional CNN for classifying the entire image would be too computationally expensive and require significant downsampling. Additionally, downsampling the image would cause discriminative details of the image to be lost. Therefore, they used a patch-based CNN to identify discriminative patches of size 400×400. The identified patches were then aggregated and used for image-level classification.

Li et al. [26] used the patch-based CNN for binary segmentation of tumor versus non-tumor voxels in the liver in an axial computed tomography image. They tested four CNNs of square patch sizes 13, 15, 17, and 19. The CNN's segmentation obtained a higher Dice score than other machine learning methods, including Adaboost, random forests, and support vector machines.

A drawback of patch-based CNNs is that the sliding window contains significant overlap when classifying neighboring pixels, resulting in many redundant computations. In addition, global context, such as the location of a pixel within the entire image, cannot be used. The patch-based CNN's features are limited to what lies in the local input region. Here lies a trade off between patch sizes. A large patch size provides more global context but less localization accuracy. A small patch size provides less global context but more

localization accuracy.

1.4 Fully Convolutional Network

The FCN takes an entire image as input and classifies every pixel in one forward pass. The FCN has no fully connected layers. Rather, it contains upsampling layers and a final $1 \times 1 \times N$ convolutional layer (N output channels) for N-class classification. Deconvolution or unpooling is used to upsample a mapping back to the original dimensions of the input image or to the dimensions of a previous layer. Deconvolutional filters can be learned through standard backpropagation, or they can be fixed (for bilinear interpolation, etc.). Deconvolution is also called transposed convolution. The encoding phase of an FCN is composed of convolutions and poolings (downsampling). The decoding phase of an FCN is composed of upsampling layers, such as deconvolutions and unpoolings.

Long et al. [27] first proposed the FCN and used it for pixel-wise image segmentation. They used a skip architecture for combining (fusing) coarse mappings from deep layers (less spatial information, more deep features) with fine mappings from shallow layers (more spatial information, less deep features) so that both local and global features are used in pixel-wise classification. Long et al. experimented with different combinations of upsampling and fusing. In their FCN-32, the final convolutional layer output is upsampled $32\times$ (deconvolution with stride 32) to the original input image's size and yields a very coarse pixel-wise segmentation. In FCN-16, the final convolutional layer

is upsampled 2× and added element-wise (fused) to the previous layer's max-pooling. This sum is then upsampled 16× to the original input image's size and yields a finer pixel-wise segmentation. Continuing in a similar fashion by fusing shallower max-pooling layer outputs with upsampled deeper layer outputs and upsampling to the original image's size, FCN-8 yields an even finer segmentation. See Figure 1.1.

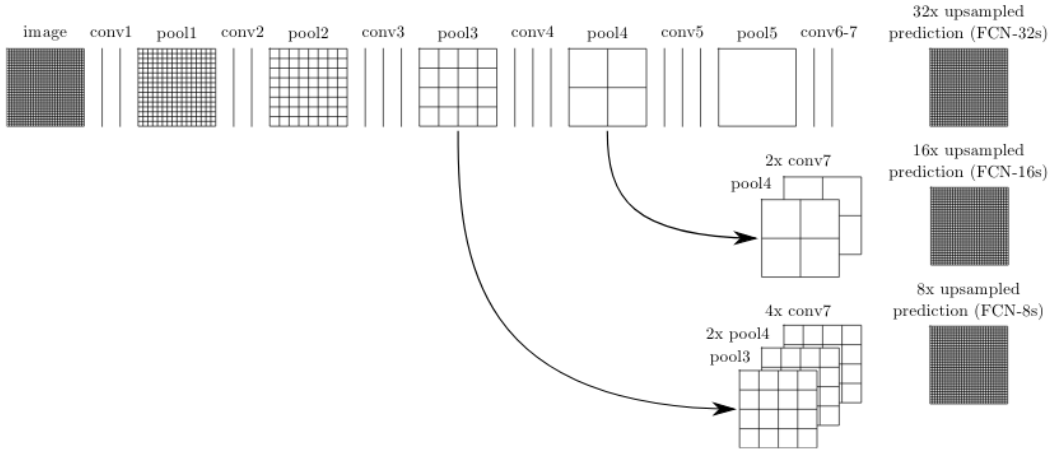

Figure 1.1: FCN architecture proposed by [27]. Fusing and upsampling of layers is shown. In FCN-32, conv7 is upsampled 32× to match the input image dimensions. In FCN-16, conv7 is upsampled 2×, fused with pool4 (through element-wise addition), and upsampled 16× to match the input image dimensions. In FCN-8, the 4× upsampled conv7, 2× upsampled pool4, and pool3 are fused and upsampled 8× to match the input image dimensions.

Ben-Cohen et al. [28] used FCNs for two separate tasks of segmenting the liver in axial CT scans and for segmenting lesions within the liver. They tested FCN-8 with 3 slices (includes above and below neighboring slices in input), regular FCN-8, and FCN-4 with 3 slices for liver segmentation. Respectively, the three networks obtained Dice scores of 0.89, 0.88, and 0.87. They also tested FCN-4 with 3 slices, FCN-8 with 3 slices, FCN-8, patch-based CNN of size 17 × 17, and a sparsity based learned dictionary for lesion segmentation.

Respectively, the models obtained true positive rates (total number of detected lesions divided by total number of known lesions) of 0.88, 0.86, 0.85, 0.85, and 0.82.

Ronneberger et al. [29] proposed the *U-Net* FCN and obtained state of the art results in segmenting ISBI 2012 electron microscopic images and ISBI 2014, 2015 light microscopic images. The novelty in U-Net was the propagation of feature channels with high global context from the contracting path to the expanding path. The contracting path (encoder) uses max-pooling for downsampling, and the expanding path (decoder) uses deconvolution for upsampling. The U-Net architecture is symmetric in that each convolutional layer in the contracting path has a corresponding convolutional layer in the expanding path. See Figure 1.2. A convolutional layer in the expanding path operates on the feature channels received from the contracting path's corresponding layer and on the previous layer's upsampled feature channels. In this way, the final convolutional layer will have information from both global and local context for pixel-wise classification. U-Net fuses feature channels from all layers through concatenation, while Long et al.'s FCN does not fuse shallow feature channels. As Long et al.'s FCN fuses shallower feature channels, improvements in segmentation performance diminish, and at a certain point performance will worsen.

Many papers that expanded upon U-Net soon followed. Cicek et al. [30] adapted U-Net's layers to their 3D counterparts to create 3D U-Net. They also modified the loss function to allow the network to train on sparsely annotated slices. They set an unlabeled pixel's loss to 0, so that the network

would only learn from labeled pixels. Milletari et al. [31] proposed *V-Net*, which has a similar architecture to 3D U-Net but with residual blocks [32], and they introduced Dice loss [equation 6]. Huang et al. [33] proposed *DenseNet* for image classification, in which a convolutional layer's input is the concatenation of all the preceding layers' outputs. Jegou et al. [34] adopted dense blocks from DenseNet in their architecture for image segmentation. In Badrinarayanan et al.'s *SegNet* [35], upsampling in decoder layers was performed using the max-pooling indices from the corresponding encoder layers and then convolved with a trainable filter to produce a dense feature map. In this method, learning an upsampling layer was not needed. In their work on *DeepLab*, Chen et al. [36] developed the *atrous convolution* as an alternative to deconvolution (uspampling) without increasing computation or the number of parameters. They also proposed *atrous spatial pyramid pooling* for capturing multi-scale context and improved object boundary detection using conditional random fields.

Due to the FCN's fixed receptive field size, large objects might be classified using local information from only part of the object, and small objects might be classified using extraneous information from outside of the object. This can result in one large object being segmented into two different objects, and one small object being classified as background, for example. These limitations are discussed by Noh et al. [37]. In their paper, they proposed *DeconvNet*, a deep deconvolutional network, that takes an instance-wise segmentation approach to address Long et al.'s original FCN's issues in handling objects of varying scales. Fu et al. [38] proposed a stacked deconvolutional network,

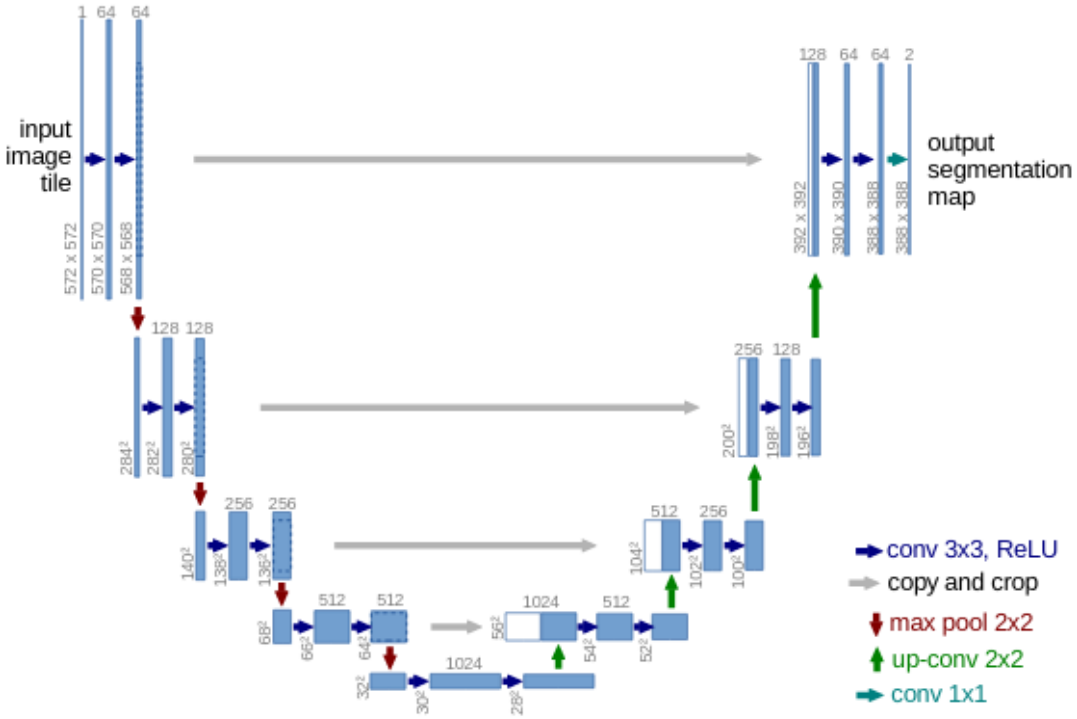

Figure 1.2: U-Net architecture proposed by [29]. Feature channels output from the contracting path (left) are concatenated to feature channels in the expanding path (right).

where shallow deconvolutional networks are stacked for improved recovery of localization information and better handling of multi-scale context.

Overall, FCNs are a powerful architecture. FCNs better maintain both global and local features for pixel-wise classification than patch-based CNNs. However, FCNs usually require more labeled data for training.

1.5 Stacked Sparse Autoencoder

An autoencoder (AE) neural network is composed of two sequential parts: an encoder and a decoder. The encoder learns efficient representations of

input data, while the decoder learns how to reconstruct input data from those representations. Training an AE is unsupervised, in that the training data does not need to be labeled. During training, the target values for computing the loss function are the input data. Therefore, the AE attempts to minimize reconstruction error.

An AE is typically composed of fully connected layers and activation layers. The size of hidden layers can be larger or smaller than the input, depending on the application at hand. An AE is often used for dimensionality reduction, in which case the encoder layers are smaller than the input and decrease in size as the network depth increases, forming a contracting path. The decoder layers then increase in size back to the original input's size, forming an expanding path. An AE can be used for denoising [39], where a clean input is reconstructed from a corrupted one. Hidden layers can also be larger than the input for many applications.

Constraints can be placed on an AE in various forms to prevent it from simply learning an identity function. The AE's architecture itself, such as hidden layer sizes, can be tuned to discover structure within the input data, like for dimensionality reduction. In addition to any L_1 and L_2 regularization, a sparsity constraint is commonly applied to help discover meaningful representations of the input data. The sparsity of an AE refers to how often a neuron in a hidden layer is "activated." A neuron is activated when its activation function outputs a high value (close to 1 for sigmoid activation, etc). Let $\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} a_j(x_i)$ be the average activation of neuron j over a dataset x with m samples, where a is the activation function. A sparsity parameter ρ is

used to constrain $\hat{\rho}_j$ to a desired value close to 0, thereby causing neuron j to activate infrequently and increasing sparsity. The value of ρ is typically set to around 0.05. An additional term Ω is added to the loss function that penalizes the model for having $\hat{\rho}_j$ deviate from ρ:

$$\Omega = \beta \sum_{j=1}^{D} \left(\rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \right) \tag{1.1}$$

where D is the number of neurons in a hidden layer and β is the sparsity regularization coefficient. Adding Ω in the loss function is what constitutes a *sparse* autoencoder. [40]

A *stacked* autoencoder is formed by training each layer on its previous layer's encoded representation. The first hidden layer is trained on input data in the typical encoder-decoder fashion. The second hidden layer is then trained on the first hidden layer's encoded features of the input data, again in the typical encoder-decoder fashion. Training the entire stacked AE continues sequentially in this way for all hidden layers. If the stacked AE is being used for a supervised classification task, then a final classification layer, such as softmax, is added. The entire network is then fine-tuned by training on the labeled data. The layers of a stacked AE are composed only of encodings, but training individual layers (before fine-tuning the entire network) uses decodings as well. Ω (Equation 1.1) can be added to the loss function when training individual layers to make each layer a sparse representation. Finally, this would result in a *stacked sparse* autoencoder (SSAE).

SSAEs have been used for both detection and segmentation tasks. The SSAE learns meaningful features and representations of input data that are

then used by a classifier. A binary classifier, such as a support vector machine (SVM), is used in detection tasks. Xu et al. [41] used an SSAE and SVM to detect the presence of breast cancer nuclei in patches of histopathological images. Similarly, Praveen et al. [42] used an SSAE and SVM to classify patches of ischemic stroke images as either lesion or normal.

In multi-class classification, including segmentation tasks, a final classification layer in the SSAE is used as previously mentioned above. Parekh et al. [43] used this architecture to segment individual voxels of multiparametric breast MRI into one of four classes: background, fat, glandular, or lesion. They also inputted features extracted from the SSAE into an SVM to classify tumors as malignant or benign.

Chapter 2

Reinforcement Learning in Medical Image Analysis

2.1 Background

The general reinforcement learning (RL) framework consists of an artificial agent that performs actions within an environment E and can be modeled as a Markov decision process (MDP). At each time point t, the agent is in a state $s \in S$ and performs an action $a \in A$ from a set of legal actions. Based on s and a, E determines the agent's transition to the next state s' and provides feedback to the agent in the form of a reward r. Transitioning to s' and receiving reward r may be stochastic and described by $p(s', r|s, a)$. The goal in RL is to have the agent learn a policy $\pi : S \rightarrow A$ that maximizes return R, which is the total cumulative reward $R = \sum_{t=0}^{T-1} \gamma^t r_t$ for an episode of length T time steps. $\gamma \in [0, 1]$ is a discount factor, where lower values emphasize immediate reward rather than long-term. An episode ends at a time step T when the agent's state reaches the terminal state. The next episode then starts independently of how the previous episode ended. Tasks may also be

continuous without a terminal state, in which case the notion of an episode does not exist and $T = \infty$. The policy π may be stochastic, and it maps an agent's perceived state to an action. [44]

A state-value function V determines the value of a state s under policy π, which is the expected total future cumulative reward the agent can attain starting from that state. $V(s) = \mathbb{E}_\pi[\sum_{t=k}^{T-1} \gamma^t r_t | s_k = s]$ at time point k. An action-value function Q (also known as the Q-function) determines the value of taking action a in state s under policy π. $Q(s,a) = \mathbb{E}_\pi[\sum_{t=k}^{T-1} \gamma^t r_t | s_k = s, a_k = a]$ at time point k. Reward r determines the immediate desirability of a state, while value determines its long-term desirability. The optimal policy π^* has corresponding optimal state-value and action-value functions $V^*(s)$ and $Q^*(s,a)$. π^* is the policy that has the highest expected cumulative reward R for all states S. That is, $V^*(s)$ is greater than all other $V(s)$ for all $s \in S$. π^* can be found by knowing either $V^*(s)$ or $Q^*(s,a)$. The Bellman equation allows us to write $Q(s,a) = E\big[r + \gamma Q(s',a')\big]$, which is the foundation of many algorithms for solving for Q^*, or an approximation. Model-based algorithms either estimate or are given the environment E and its transition property $p(s',r|s,a)$. Model-free algorithms learn a policy without any knowledge or estimation of $p(s',r|s,a)$. [44]

The exploration versus exploitation trade-off is encountered when training and agent. In order to discover new actions with a possibly higher reward, an agent may need to take low-reward actions that would otherwise be ignored when trying to maximize reward based on the current policy. This is the notion of exploration. In contrast, the agent should also take actions that have been

learned in its current policy as being high-reward in order to maximize return. This is the notion of exploitation. The proper balance between exploration and exploitation is a fundamental dilemma in RL. An imbalance toward exploration can result in the agent never converging to a high-value policy, while an imbalance toward exploitation can result in the agent finding a sub-optimal policy. Algorithms for learning a policy can be on-policy or off-policy. An on-policy algorithm improves the current policy by performing actions that follow the same policy. An off-policy algorithm performs actions that are different from the policy being learned. [44]

2.2 RL in Medical Image Analysis

RL methods in medical image analysis have been used for tasks such as landmark localization, image segmentation, and image registration. Sahba et al. published several works in image segmentation using RL techniques without deep learning. In [45], they segmented the prostate in ultrasound images using Q-Learning. Their method divides an image into sub-images. Within each sub-image, the agent performs actions such as changing the thresholding value and the size of morphological openings. The reward is the change in quality measure of the resulting segmentation. In [46], they incorporated an opposition-based RL scheme [47] for improving learning convergence speed in their previous work. In [48], they demonstrated their methods in transrectal ultrasound (TRUS) images of the prostate.

Deep RL techniques for anatomical landmark localization were a focus of Ghesu et al.'s work. In [49], a deep Q-network (DQN) was used in an

artificial agent to find the optimal path to a desired landmark in 2D magnetic resonance, 2D ultrasound, and 3D CT images. In their framework, the state s is an agent's region of interest (cropping) of an image. The action set A includes discrete movements in the agent's region of interest by one pixel (up, down, left, etc.). The reward r is the change in distance to the target landmark after one move. Their method outperformed previous exhaustive-search techniques in terms of both accuracy and speed. Their method could also detect the absence of a landmark. In [50], they extended their method to multi-scale image representations and compared it to other approaches for landmark detection. This work considered eight different landmarks in 1,487 3D CT scans. In [51], they further developed their methods to locate structures that are not within the field of view of the image. Model evaluation was performed on 5,043 3D CT volumes and considered 49 different landmarks.

Expanding on Ghesu et al.'s work, Alansary et al. [52] evaluated multiple DQN architectures for anatomical landmark localization, including standard DQN [53, 54], double DQN [55, 56], duel DQN [57], and duel double DQN. These modified DQN architectures were developed to improve the DQN's training stability and convergence. Alansary et al. also proposed multi-scale hierarchical search steps, where the agent's field of view and step size decrease as the search converges on the target landmark. They evaluated their methods on three landmarks in 72 3D fetal head ultrasound scans and concluded that selecting the best DQN architecture is environment-dependent. A multi-agent framework for locating multiple landmarks in one search was later proposed by Vlontzos et al [58]. A multi-agent model is computationally more efficient

than multiple single-agent models because only one DQN is used in the multi-agent setting.

RL techniques for image registration using DQNs were proposed in [59, 60, 61]. Liao et al. [59] trained a DQN to find a rigid-body registration transformation through sequential translations ($\pm$1mm) and rotations ($\pm 1^\circ$). Their method is supervised in the sense that the learned transformation is compared to a ground truth transformation when calculating reward. Similarly, Ma et al. [60] trained a modified dueling DQN [57] to obtain a rigid-body registration transformation. In addition, their approach keeps a history of actions to help prevent the agent from oscillating around certain image positions. Krebs et al. [61] generalized Liao et al.'s methods to non-rigid (deformable) registration.

2.3 Q-Learning

Many of the deep reinforcement learning methods used in the works mentioned above use deep Q-learning. Q-learning [62] is an off-policy model-free algorithm for learning an action-value function $Q(s, a)$ that estimates the optimal $Q^*(s, a)$. By knowing the optimal Q^*, the optimal policy π^* would be to perform the action a that maximizes $Q^*(s, a)$ in every state $s \in S$: $a = \arg\max_a Q^*(s, a)$. Algorithm 1 describes the Q-learning algorithm. The algorithm's ϵ-greediness sets a balance between exploration and exploitation. A smaller ϵ encourages exploitation, and a larger ϵ encourages exploration. While a_t can be chosen either greedily (Line 9) or randomly with probability ϵ (Line 11), Q is updated in Line 14 using $\max_a Q(s', a)$, which strictly chooses the greedy action. The fact that Q is updated using a different policy is what

makes Q-learning an off-policy algorithm.

Algorithm 1: Q-Learning algorithm with ϵ-greediness as described in [44].

Input: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Result: $Q(s, a)$: an approximation of $Q^*(s, a)$
Initialize $Q(s, a)$ for all $s \in S$ and $a \in A$ arbitrarily except that $Q(\text{terminal state}, \cdot) = 0$
for *each episode* **do**
    $t = 1$
    Initialize a starting state s_1
    while *s is not terminal* **do**
        Generate $z =$ random uniform number between 0 and 1
        if $z > \epsilon$ **then**
            // Choose a_t using the policy derived from Q_t
            $a_t = \arg\max_a Q_t(s_t, a)$
        else
            Randomly choose action a_t from A
        end
        Perform action a_t, observe reward r_t and the next state s_{t+1}
        $Q_{t+1}(s, a) = Q_t(s, a) + \alpha[r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$
        $t = t + 1$
    end
end

In deep Q-learning [53, 54], a neural network, called a deep Q-network (DQN), is trained to generate $Q(s, a)$. The DQN takes state s as input and computes an action-value $Q(s, a)$ for each of the possible actions $a \in A$. Typically, the final layer in a DQN is fully connected with n outputs for n possible actions. The loss function $L = [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2$, where s' is the next state, is computed using two separate DQNs for improved training stability: a target DQN T and a prediction DQN P. Both T and P have the same exact architecture. T outputs the target value $r + \gamma \max_{a'} Q(s', a')$, and

P outputs the predicted value $Q(s, a)$. P's weights are updated every iteration. T's weights are updated every N iterations to P's weights but are otherwise fixed. Training samples are taken from an experience replay buffer containing an agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$.

Mnih et al. first proposed the DQN with experience replay to play seven Atari games in [53]. In this work, they used a single neural network (CNN) to compute target and prediction values. Their models performed similar to or better than expert humans in certain games and outperformed previous RL methods. However, their models under-performed an expert human in games that required a long-term strategy. Mnih et al. revisited their work in [54] and formally coined the term "DQN." They proposed using separate target and prediction networks for better training stability and evaluated their models on 49 Atari games. Similar to their previous work, their models generally outperformed an expert human except in games that required a long term strategy. Mnih et al.'s methods in DQNs have been extensively used by researchers in many fields, such as in the previously mentioned works in medical image analysis.

Chapter 3

Segmentation Models

3.1 Introduction

Deep learning algorithms are beginning to emerge in radiological applications for segmentation and classification of different diseases [15, 29, 63]. These deep learning algorithms have the ability to learn different features by using the inherent tissue contrasts from multiparametric magnetic resonance imaging (mpMRI). Multiparametric deep learning (MPDL) and radiomics were previously applied to the assessment of brain tumors and breast cancer [43, 64, 65, 66].

In this work, I developed novel MPDL segmentation models based on mpMRI and applied them to limb girdle muscular dystrophy 2I (LGMD2I). LGMD2I is a genetic disease that causes normal muscle to be replaced by fat over time, and it typically affects the muscles in the shoulder and pelvic girdle [67]. Areas where normal muscle have been replaced by fat are "fatty infiltrated." While there is no cure, drugs are being tested to slow the progression of the disease in patients. Automatic segmentation of the disease in MRIs

can help track the disease's progression and determine whether treatment is effective over time. There exists no prior work in automatic segmentation of muscular dystrophy. I focused on muscle groups in the thighs, in which separating fat, fat-infiltrated muscle, and healthy muscle tissue can be challenging. Specifically, the MPDL models classify voxels as either background, muscle, bone, fat, or fat infiltration. I compare the performance of two separate MPDL models: a patch-based convolutional neural network (CNN) (described in Section 1.3) and a stacked sparse autoencoder (SSAE) (described in Section 1.5).

An mpMRI is composed of multiple registered sequences obtained from different imaging parameters, such as T1WI, T2WI, DWI, etc. Therefore, one 2D slice in an mpMRI has multiple channels (one for each imaging parameter). A tissue signature (TS) is a vector composed of gray level intensities from a voxel position across all the channels in a 2D mpMRI slice. In general, a TS for a voxel position is defined as follows:

$$TS^{(\tau)} = [T_1, T_2, DIXON, DWI, \dots] \tag{3.1}$$

where τ is the tissue type, and each element in the vector is the intensity at the voxel position in each channel. A TS can take on a second form, in which each element in the TS is a square patch of intensities at the same location in each channel. The CNN trains on patch-TS, while the SSAE trains on vector-TS.

Current state-of-the-art methods in medical image segmentation use fully convolutional networks (FCN) such as U-Net [29]. FCNs usually require fully labeled training data, where every pixel in an image is labeled, and are less

applicable to partially labeled datasets. Significant time and effort are required to label every pixel of an image in an unlabeled dataset, even more so if there are several possible classes a label can take. This makes FCNs less applicable to newly obtained (unlabeled) datasets. Therefore, the MPDL models that train on tissue signatures are used in this work instead.

3.2 Dataset

The dataset used in my experiments was composed of whole body mpMRIs of 19 patients and six normal (healthy) subjects. Sequences available for use were Dixon, T1WI, and T2WI. A Dixon sequence acquires four images (channels) per 2D slice in a whole body volume: in phase, out of phase, fat, and water. The four Dixon images are registered upon acquisition. I manually registered T1WI and T2WI sequences to the Dixon images in order to incorporate all six images (one T1, one T2, four Dixon) in tissue signatures.

I created two datasets used in this study by manually labeling patches of different tissues in thigh slices of patients' MRIs. One dataset was composed of tissue signatures only of Dixon images, in which one training sample is $TS^{(\tau)} = [Dixon_{in}, Dixon_{out}, Dixon_{fat}, Dixon_{water}]$. The other dataset was composed of tissue signatures of Dixon, T1WI, and T2WI images, in which one training sample is $TS^{(\tau)} = [Dixon_{in}, Dixon_{out}, Dixon_{fat}, Dixon_{water}, T_1, T_2]$. The datasets were obtained from a small subset of thigh slices of patients with only a small fraction of each slice being labeled. Of the slices used in the Dixon dataset, an average of 8.4% of voxels were labeled per slice.

3.3 Network Architectures

The two MPDL architectures used were patch-CNNs and SSAEs. The CNN consists of four convolutional layers with 128, 64, 32, and 16 filters respectively of size 3×3, followed by a fully connected layer and a softmax layer. The cross entropy loss function was used. Training was performed with the Adam optimizer with a learning rate of 0.001, $momentum = 0.9$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a minibatch size of 1024. The CNN takes a square patch of dimensions $k \times k \times n$ as input, where k is the square patch size and n is the number of images (channels) in a 2D slice. During inference, the CNN classifies every voxel in a sliding window fashion and classifies the patch's center pixel's tissue type. More details on the patch-based CNN are provided in Section 1.3. I evaluated CNNs with square patches of size $k = 5, 7, 9, 11$.

The SSAE consists of fully connected layers and a final softmax layer. Cross entropy loss was used. The SSAE's general training procedure is described in Algorithm 2. Specifically, hidden layer i's encoding output $z_i \in \mathbb{R}^{h_i}$ of an input sample $x \in \mathbb{R}^n$ is $z_i = f_i(x) = a(W_i{}^{(1)}x + b_i{}^{(1)})$, where a is the activation function (nonlinearity), $W_i{}^{(1)} \in \mathbb{R}^{h_i \times n}$ is the encoder weight matrix, and $b_i{}^{(1)} \in \mathbb{R}^{h_i}$ is the encoder bias vector. A hidden layer's decoding output $\hat{x} \in \mathbb{R}^n$ is an estimate of its input $x \in \mathbb{R}^n$ and is only used when training that individual layer. $\hat{x} = g_i(f_i(x)) = a(W_i{}^{(2)}f_i(x) + b_i{}^{(2)})$, where a is the activation function, $W_i{}^{(2)} \in \mathbb{R}^{n \times h_i}$ is the decoder weight matrix, and $b_i{}^{(2)} \in \mathbb{R}^n$ is the decoder bias vector. The loss function for training a layer is the mean squared error between x and $\hat{x}$ plus L_2 and sparsity regularization terms. Training data x is the input for training the first hidden layer. Subsequent

hidden layers train on the previous layer's encoding as input: $z_i = f_i(z_{i-1})$ and $\hat{z}_{i-1} = g_i(z_i)$. Training was performed with the sigmoid function as the activation function, a sparsity regularization coefficient of 4, a desired sparsity proportion of 0.25, and an L_2 regularization coefficient of 0. More details on the SSAE are provided in Section 1.5.

Algorithm 2: Training MPDL SSAE

Input: Training data X (tissue signatures)
Hidden layer sizes $[h_1, h_2, ...h_m]$ (m hidden layers)
Result: A trained SSAE

```
Train hidden layer 1 of size h_1 with X as input
Compute encoded features z_1 = f_1(X)
for i = 2 to m do
    Train hidden layer i of size h_i with z_{i-1} as input and target
    Compute encoded features z_i = f_i(z_{i-1})
end
Train softmax layer with z_m as input and using X's labels
Stack all of the trained hidden layers and softmax layer into one model
Train whole model on X
```

Each hidden layer is trained individually in order of first to last. In training the first hidden layer, both the input and target values are a tissue signature. In training subsequent hidden layers, the previous layer's features are used as input and target values. The SSAE takes a length n vector of intensities of a single voxel position over n channels in one 2D slice. During inference, the SSAE iterates over each individual voxel in the 2D slice and classifies the voxel's tissue type.

3.4 Evaluation Methods

I evaluated segmentation performance of MPDL CNN and SSAE in two datasets. One dataset was composed only of the Dixon sequences (4 channels), and the other was composed of Dixon, T1, and T2 sequences (6 channels). In both datasets, I evaluated four CNNs with patch sizes 5, 7, 9, and 11, and two SSAEs of hidden layer sizes 10-10 and 10-10-5-10-10.

To evaluate the MPDL segmentation, I developed an algorithm that uses the Eigenimage (EI) filter [68], a semi-supervised segmentation method, to obtain ground truth segmentations of muscle, fat, and fatty infiltrated tissue of the thighs in the Dixon MRI. See Algorithm 3. Using this algorithm to generate ground truth segmentations of the thighs was necessary. Segmentations are obtained by thresholding the output EI for each slice, which would take a prohibitively long time for a human to manually perform. Dice similarity was calculated between the model output and the EI output (ground truth) as a metric for model accuracy.

Algorithm 3: Generating Eigenimage ground truth segmentations. Input ROIs must be manually created for a single slice.

Input: Single Dixon 2D slice (4 channels)
Muscle ROI, fat ROI, fat inf ROI for the single Dixon slice
Dixon 3D volume
start = thigh start slice number
end = thigh end slice number
Result: Eigenimage ground truth segmentations for all thigh slices in 3D volume

```
Train Eigen filter on ROIs in the single Dixon slice
for i = start to end do
    Apply trained Eigen filter to Dixon slice i
    Extract filter's muscle, fat, and fat inf outputs
    Binarize (threshold) each output using Otsu's method to obtain
     final segmentation for the slice
end
```

3.5 Results

3.5.1 Dixon-Only Sequences

The Dice metrics of the CNN Patch Size 5 and SSAE segmentations for the normal volunteers are shown in Table 3.1 and Table 3.2. Dice metrics for all models in all patients are shown in Table 3.3. Figure 3.1 shows an example of the segmentation and thresholded EI of a slice.

3.5.2 Dixon, T1, and T2 Sequences

Dice metrics for all models in all patients using Dixon, T1, and T2 images are shown in Table 3.4.

Dixon-only mpMRI of Normals

Model without fat inf	Muscle Dice	Fat Dice
CNN 5	0.85 ± 0.07	0.77 ± 0.13
SSAE 10-10	0.90 ± 0.05	0.80 ± 0.17

Table 3.1: Dice scores (mean $\pm$ stdev) between the MPDL segmentation and Eigenimage segmentation of the thighs in six normal subjects without fat infiltration as a possible label.

Dixon-only mpMRI of Normals

Model with fat inf	Muscle Dice	Fat Dice
CNN 5	0.86 ± 0.06	0.47 ± 0.13
SSAE 10-10	0.88 ± 0.06	0.74 ± 0.21

Table 3.2: Dice scores (mean $\pm$ stdev) between the MPDL segmentation and Eigenimage segmentation of the thighs in six normal subjects with fat infiltration as a possible label.

Dixon-only mpMRI of Patients

Model	Muscle Dice	Fat Dice	Fat Inf Dice
CNN 5	0.78 ± 0.13	0.55 ± 0.15	0.60 ± 0.07
CNN 7	0.74 ± 0.11	0.70 ± 0.12	0.56 ± 0.08
CNN 9	0.75 ± 0.13	0.65 ± 0.12	0.55 ± 0.09
CNN 11	0.73 ± 0.14	0.69 ± 0.11	0.54 ± 0.08
SSAE 10-10	0.90 ± 0.06	0.87 ± 0.10	0.74 ± 0.09
SSAE 10-10-5-10-10	0.85 ± 0.06	0.91 ± 0.04	0.70 ± 0.09

Table 3.3: Dice scores (mean $\pm$ stdev) between the MPDL segmentation and Eigenimage segmentation of the thighs in 19 patients. Input mpMRI was composed only of Dixon images (four channels).

Dixon, T1, T2 mpMRI of Patients

Model	Muscle Dice	Fat Dice	Fat Inf Dice
CNN 5	0.74 ± 0.13	0.78 ± 0.08	0.56 ± 0.07
CNN 7	0.75 ± 0.12	0.79 ± 0.07	0.53 ± 0.07
CNN 9	0.75 ± 0.11	0.77 ± 0.08	0.53 ± 0.08
CNN 11	0.73 ± 0.12	0.79 ± 0.08	0.52 ± 0.08
SSAE 10-10	0.85 ± 0.12	0.85 ± 0.05	0.72 ± 0.08
SSAE 10-10-5-10-10	0.87 ± 0.11	0.87 ± 0.05	0.77 ± 0.08

Table 3.4: Dice scores (mean $\pm$ stdev) between the MPDL segmentation and Eigenimage segmentation of the thighs in 19 patients. Input mpMRI was composed of Dixon, T1, and T2 images (six channels).

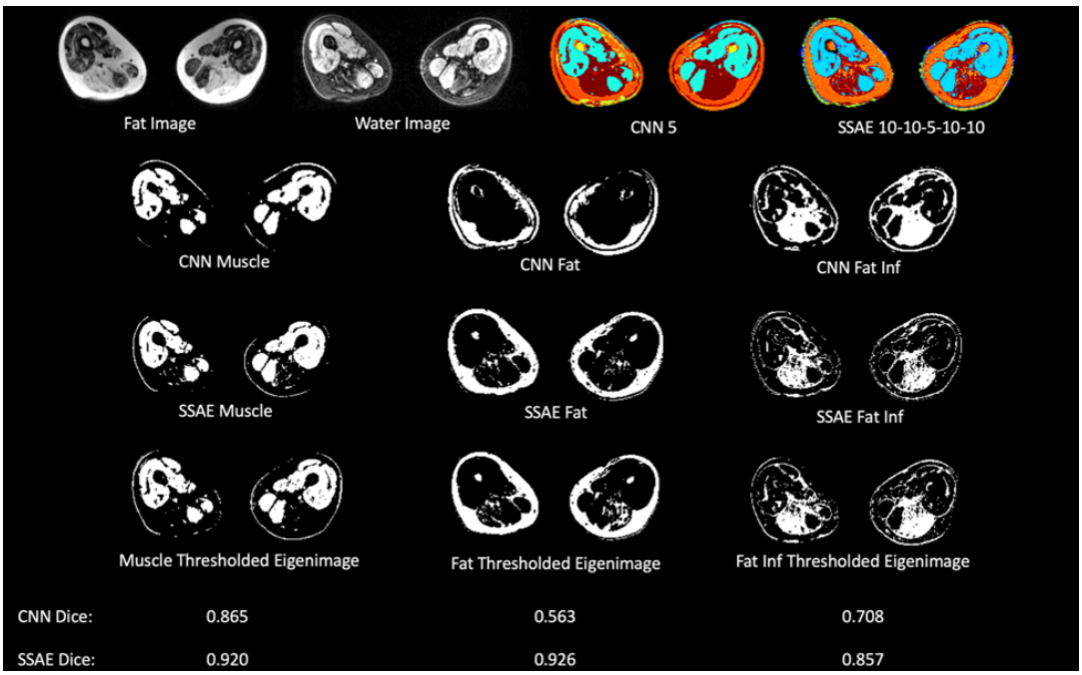

Figure 3.1: Segmentation results of a slice. The output segmentations of the CNN patch size 5 and the SSAE 10-10-5-10-10 are shown. The left, middle, and right columns contain muscle, fat, and fat infiltration segmentations respectively. The thresholded Eigenimages are treated as the ground truth.

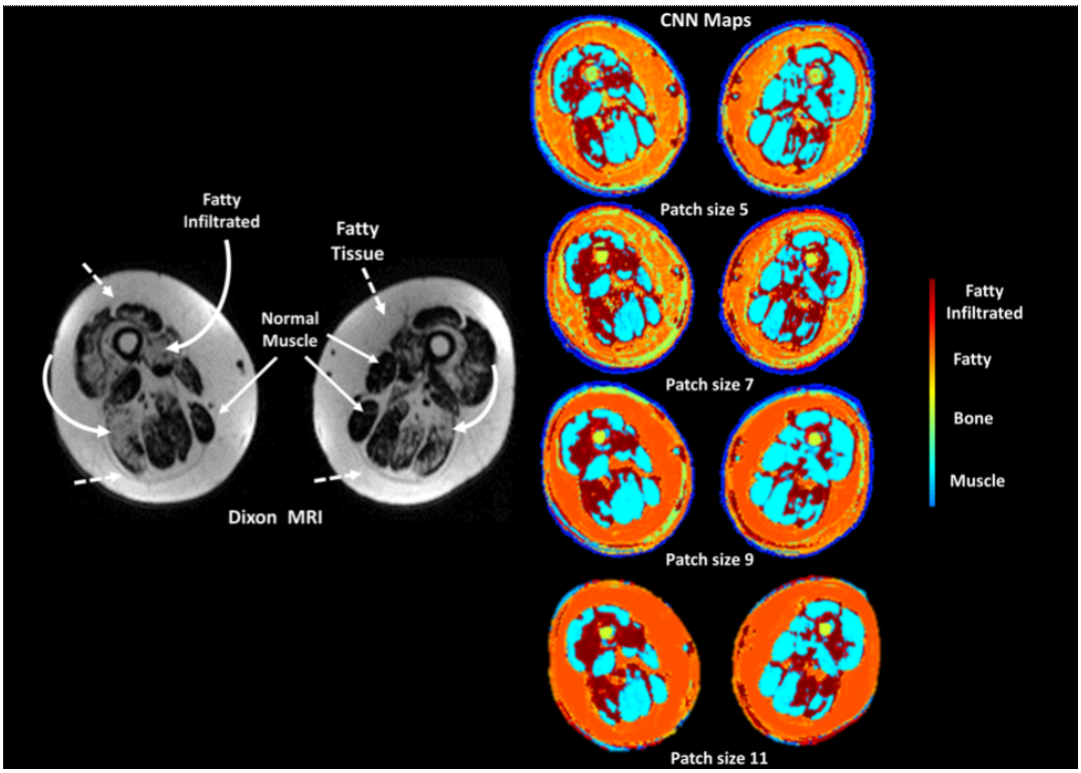

Figure 3.2: Demonstration of the muscle tissue segmentations from the MPDL tissue signature model and CNN segmentation map. Left) Example Dixon MRI on a LGMD2I patient with the different tissue types labeled. The straight arrows show normal muscle. The curved arrows show the fatty infiltrated muscle, and the dotted arrows show the fatty tissue. Right) The CNN maps using different input patch sizes with the color coding shown to the right of the images.

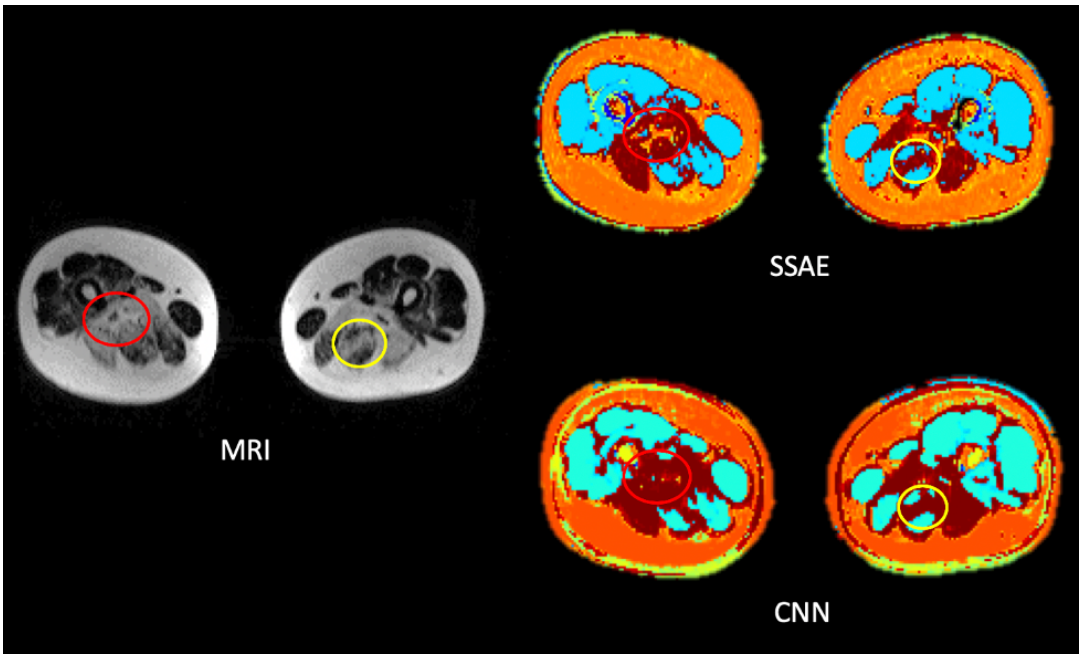

Figure 3.3: Comparison of SSAE segmentation and CNN segmentation. The SSAE produces a higher-resolution segmentation than the CNN. In the red circle, there is a small region of fat (smooth white color) surrounded by fat infiltration (grainy gray color). The SSAE clearly delineates this region of fat surrounded by fat infiltration, while the CNN classifies the whole region as fat infiltration. The yellow circle contains a region of muscle and fat infiltration. The CNN produces a more blurred, lower resolution segmentation than the SSAE. However, both models capture the region's general features.

3.6 Discussion

Thresholded EIs were used as approximations of ground truth segmentations (generated from Algorithm 3). However, there is no optimal threshold for the EIs. Increasing the threshold to include all voxels of a tissue will also include voxels of other undesired tissue. Similarly, decreasing the threshold to exclude other tissue will also exclude some of the desired tissue. Figure 3.1 shows an example where the muscle EI also contains undesired skin voxels. I therefore used Otsu's method to binarize the EIs and visually determined that the segmentations were close approximations. In addition, knowing what the ground truth should be for a region in the image can be a challenge on its own. The transitions from muscle to fat infiltration and from fat infiltration to completely fat-replaced are not clear-cut. When a region is undergoing a transition, it can be difficult to determine whether to classify the region as its original state or as its new state.

Overall, the SSAE produces a sharper segmentation and performs better than the CNN (see Figure 3.3). This can be explained by the fact that adjacent patches in the CNN contain much overlap, causing adjacent voxels to be classified as the same class and boundaries to be blurred. Instead, the SSAE's segmentation is not influenced by adjacent voxels. However, a sharper segmentation does not necessarily mean a "better" segmentation. The SSAE might be more accurate locally on a voxel-by-voxel basis, but the CNN might be more accurate regionally. The local versus region-wide accuracy tradeoff is especially visible among CNNs of different sized patches in Figure 3.2. There is not necessarily a "right" or "wrong" segmentation in regard to sharpness.

For example, there could be a thin sliver of fat infiltration within muscle that is emphasized in the CNN segmentation and understated in the SSAE segmentation. The opposite, more likely case could also occur where the CNN misses the sliver and the SSAE clearly identifies it. If the sliver is that thin, it is not necessarily right or wrong to identify it or to ignore it in the output segmentation. The radiologist can view the segmentations of the different models knowing the models' output characteristics to get a fuller sense of the image.

CNN patch sizes ranged from from 5x5 to 11x11. Smaller patch sizes resulted in sharper segmentations with different classes often scattered within a region. Larger patch sizes resulted in larger, smoother regions of a single class without scattered voxels from other classes within those smooth regions. The tradeoff with patch size is individual-voxel classification accuracy versus region-wide classification accuracy, as explained in the previous paragraph. Figure 3.2 shows an example of this relationship.

Additionally, I trained models on tissue signatures without fat infiltration (fat infiltration not being a possible output class) to compare performance on normal subjects. Results are reported in Table 3.1 and Table 3.2. Ideally, both versions of the models (with versus without fat infiltration as a possible output class) would output the same segmentation on normal subjects. This was not the case, as models with fat infiltration as a possible output classified certain voxels as fat infiltration. Both versions performed about the same when classifying muscle. However, there was a significant difference in accuracy when classifying fat, especially for the CNN. The SSAE's outperformance of

the CNN is again evident in these results. Still, the models obtained adequate segmentation accuracy of normal subjects, which further suggests that the models correctly learned the different tissue types.

Incorporating T1 and T2 images in the mpMRI improved segmentation results, as shown in Tables 3.3 and 3.4. CNN models that used Dixon, T1, and T2 mpMRIs obtained significantly higher Dice scores when classifying fat than did CNN models that used only Dixon mpMRIs. The rest of the Dice scores between the two mpMRIs were similar.

The efficacy of training deep learning models using tissue signatures is noteworthy. There is enough contrast in the mpMRI sequences for the MPDL models to discern different tissue types using only local spatial information. Adding more images from different acquisition parameters can provide sufficient contrast between tissue types. This is demonstrated by the fact that results improved when incorporating T1 and T2 sequences with the Dixon sequences. Even with a relatively small training dataset, where only small portions of slices were labeled, models performed well and produced accurate segmentations.

Chapter 4

Reinforcement Learning for Landmark Localization

4.1 Introduction

Reinforcement learning (RL) methods have recently been used for locating anatomical landmarks in medical images, as described in Section 2.2. Locating a landmark complements segmentation tasks in several ways. For example, if an anatomical object of interest can be located in both a patient's and a known healthy subject's images, then automatically detecting any abnormalities in the patient can be confined to a specific region. A small, precise area of interest contains less noise and extraneous objects than would a large area. Therefore, a model would be more likely to produce accurate segmentations. In addition, landmark localization can be used to improve segmentation algorithms that do not consider an object's broader spatial location within the entire image. Since anatomical objects are often located in a specific part of an image, models that learn such spatial features may better discern different objects. However, these models, such as a fully convolutional network, usually require

fully-labeled training images, which are often not available in medical image datasets in a hospital setting. This is a reason the MPDL models were used in Chapter 3. Spatial features acquired through landmark localization could be used to augment the MPDL models' input features and ultimately improve segmentation performance. For example, if there are two different anatomical objects with similar intensities (low contrast), an MPDL model might incorrectly classify them as the same object. However, providing the objects' spatial features, such as location, may help the MPDL model distinguish the two objects.

As an initial step in this direction, I explored RL techniques for landmark localization in various MRI sequences. More specifically, I showed that one RL model can locate different landmarks in 2D and 3D images of different anatomical environments of different imaging parameters in a multitask modality invariant deep reinforcement learning (MIDRL) framework. I evaluated two RL models: a single agent model [52] operating in 2D slices of breast MRI, prostate MRI, and whole body MRI, and a multi-agent model [58] operating in 3D volumes of whole body MRI.

The 2D single agent model was trained to locate six different anatomical structures in 2D slices throughout the body, including the chest (heart, breast), abdomen (kidney, prostate), pelvis (lesser trochanter), and lower extremity (knee) using T1-weighted (T1WI), T2-weighted (T2WI), Dynamic Contrast Enhanced (DCE), Diffusion Weighted Imaging (DWI) with Apparent Diffusion Coefficient (ADC) mapping, and DIXON MRI sequences obtained from multiparametric breast, prostate, and whole body (WB) MRI acquisitions.

The 3D multi-agent model was trained to locate the heart, kidney, trochanter, and knee landmarks (four different anatomical structures) in 3D whole body MRI of Dixon, T1WI, and T2WI sequences.

4.2 Dataset

The clinical data consisted of 57 mpMRIs. These datasets included 25 whole body (44%), 24 (42%) breast and eight (14%) prostate mpMRIs. The patient population and image acquisition parameters have been detailed in the following subsections.

4.2.1 Breast mpMRI

The breast mpMRI dataset consisted of 12 women with malignant lesions imaged at 1.5T at two timepoints for developing treatment response metrics. The mpMRI was acquired before and after the first cycle of chemotherapy. The imaging protocol included sagittal fat suppressed (FS) T2WI spin echo (TR/TE=5700/102ms) and fast spoiled gradient echo images (FSPGR) T1WI (TR/TE = 200/4.4 ms) with field of view (FOV)=18cm$\times$18cm, matrix=256$\times$192, slice thickness (ST): 4 mm, 1mm gap). Dynamic contrast enhanced (DCE) FSPGR T1WI (TR/TE=20/4ms, matrix=512$\times$160, ST: 2 mm, 3-4 phases after injection) was obtained after intravenous administration of GdDTPA contrast agent (Omniscan, Amersham Health, 0.2 mL/kg (0.1 mmol/kg)). The contrast agent was injected over 10 seconds with MR imaging beginning immediately after completion of the injection. The contrast bolus was followed by a 20cc saline flush. Total scan time was less than 20 minutes. In summary, five

different images – T1WI, T2WI, Pre-contrast DCE, Post-contrast DCE, and subtraction DCE were evaluated for the detection of nipple within the breast images.

4.2.2 Prostate mpMRI

The prostate mpMRI dataset consisted of 8 patients with prostate cancer. The mpMRI parameters were were T2WI(TR/TE-3000/30, FOV=240cm$\times$240cm, Matrix=256$\times$256, Slice thickness(ST)=3mm, NEX=2), Trace DWI (TR/TE=2000-3000/70-42, FOV=221$\times$250, Matrix=256$\times$256, ST=3mm, b-values = 0, 400, 800). ADC maps were constructed using a monoexponential equation. The imaging parameters of T2WI and ADC map were evaluated in this study for localization of prostate within the images.

4.2.3 Whole body mpMRI

The WB mpMRI dataset consisted of 25 subjects acquired using the imaging protocol that scanned from the shoulders to the lower mid calf and described in [69]. Of the 25 subjects, there were 19 patients with muscular dystrophy and six normal volunteers. The imaging parameters of T1WI, T2WI and DIXON weighted images were acquired at 3T and evaluated in this study for detection of heart, left kidney, left trochanter, and left knee.

4.3 Deep Q-Learning Framework

Deep RL is an emerging area of active research that has produced excellent results across diverse domains [53, 54, 70, 71, 72]. Briefly, RL deals with an

artificial agent that is learning to understand its environment, while attempting to maximize the cumulative award associated with a set of complex tasks [44]. Every action of the artificial agent is evaluated based on its contribution to the final cumulative reward. In deep RL, deep learning architectures such as convolutional neural networks (CNN) are used to identify the current state and predict the best possible action. These networks are known as deep Q-networks (DQN) and estimate the Q-function. The goal of deep RL algorithms in landmark localization is to learn to locate different anatomical landmarks with high accuracy, computational efficiency, and robustness.

The Q-learning algorithm attempts to learn a policy for maximizing the expected total future reward. Policy π dictates what action a the artificial agent takes in state s. The DQN approximates the optimal state-action value function (Q-function)

$$Q(s,a) = \max_{\pi} E\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi\right] \tag{4.1}$$

where r_t is the reward at time step t and $\gamma \in [0,1]$ is the discount factor. Using the Bellman equation, $Q(s,a)$ can be solved iteratively in the form of

$$Q_{i+1}(s,a) = E\left[r + \gamma \max_{a'} Q_i(s',a')\right] \tag{4.2}$$

where i is the iteration, s' is the next state, and a' is the next action. See Section 2.3 for more information on Q-learning.

Both 2D and 3D models used in the framework are DQNs. In the 2D model, the input state s to the DQN is a sequence of areas (frames) of the 2D image cropped by the agent's bounding box. More specifically, one input sample

with a sequence of length C frames has dimensions $C \times H \times W$, where C is the number of channels and $H \times W$ are the dimensions of the cropped areas. The 3D model's input state s is analogous to the 2D case but with a 3D bounding box. A multiscale technique is adopted, in which cropped areas are obtained by sampling the 2D image with a stride. For example, a multiscale state is always $H \times W$ in size, but it can be sampled from a larger area of the image. The sampling stride is initially large to cover a larger area of the image at a lower resolution. When the agent converges (oscillates around a point), both the sampling stride and action step size decrease, and the agent sees a smaller area of the image at a higher resolution. Changing the agent's field of view and step size in this way results in faster, more accurate convergence to the target landmark.

The training process uses a target network, DQN_T, and a policy network, DQN_P, both with the same architecture. Upon initialization, both networks have the same weights. During one training step, the input to DQN_P is state s, and the input to DQN_T is the next state s'. DQN_P outputs the predicted Q-value $Q^*(s, a)$, and DQN_T outputs $Q^*(s', a')$. The target Q-value is $r + \gamma Q^*(s', a')$. After computing the loss function, DQN_P's weights are updated on each step using stochastic gradient descent. DQN_T's weights are only updated to DQN_P's weights every N steps. This is done to stabilize training.

Training samples are selected from an experience replay buffer. During training, an experience replay buffer is populated with the state, action, reward, and resulting state $[s, a, r, s']$ from steps taken over many episodes. Steps are taken according to an ϵ-greedy policy, where an action is taken uniformly

at random with probability ϵ at each step. Otherwise, the action with the highest reward is chosen.

Overall, the RL framework for the 2D model is composed as follows. The environment is a 2D image in which the agent is a square bounding box that has four possible actions: $a \in$ {move up, move down, move left, move right}. The state is the sequence of cropped areas as previously described above. The reward is the difference between the Euclidean distance from the target landmark to the agent before and after the agent takes a step in a certain direction. The reward is positive if the agent moves closer to the landmark and negative if it moves farther. The landmark's location is its (x, y) coordinate in the image, and the agent's location is the bounding box's center (x, y) coordinate. The 3D model's framework is the same as the 2D case but with a third dimension for operating in a volume. In addition, the 3D model is a multi-agent model with four agents for locating four different landmarks.

4.4 Training Details

The 2D DQN is composed of four 2D convolutional layers with maxpooling and PReLU activations and four fully connected layers with leaky ReLU activations. See Figure 4.2.

Stochastic gradient descent is performed using the Adam optimizer with learning rate $= 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.001$. The Huber loss function is used. Training samples have a frame history of length $C = 4$, cropping of size $H \times W = 45 \times 45$, and batch size of 48. Discount factor $\gamma = 0.9$. Reward r is clipped to between -1 and 1. $\epsilon = 1$ at epoch 1 decreases to $\epsilon = 0.1$ by epoch

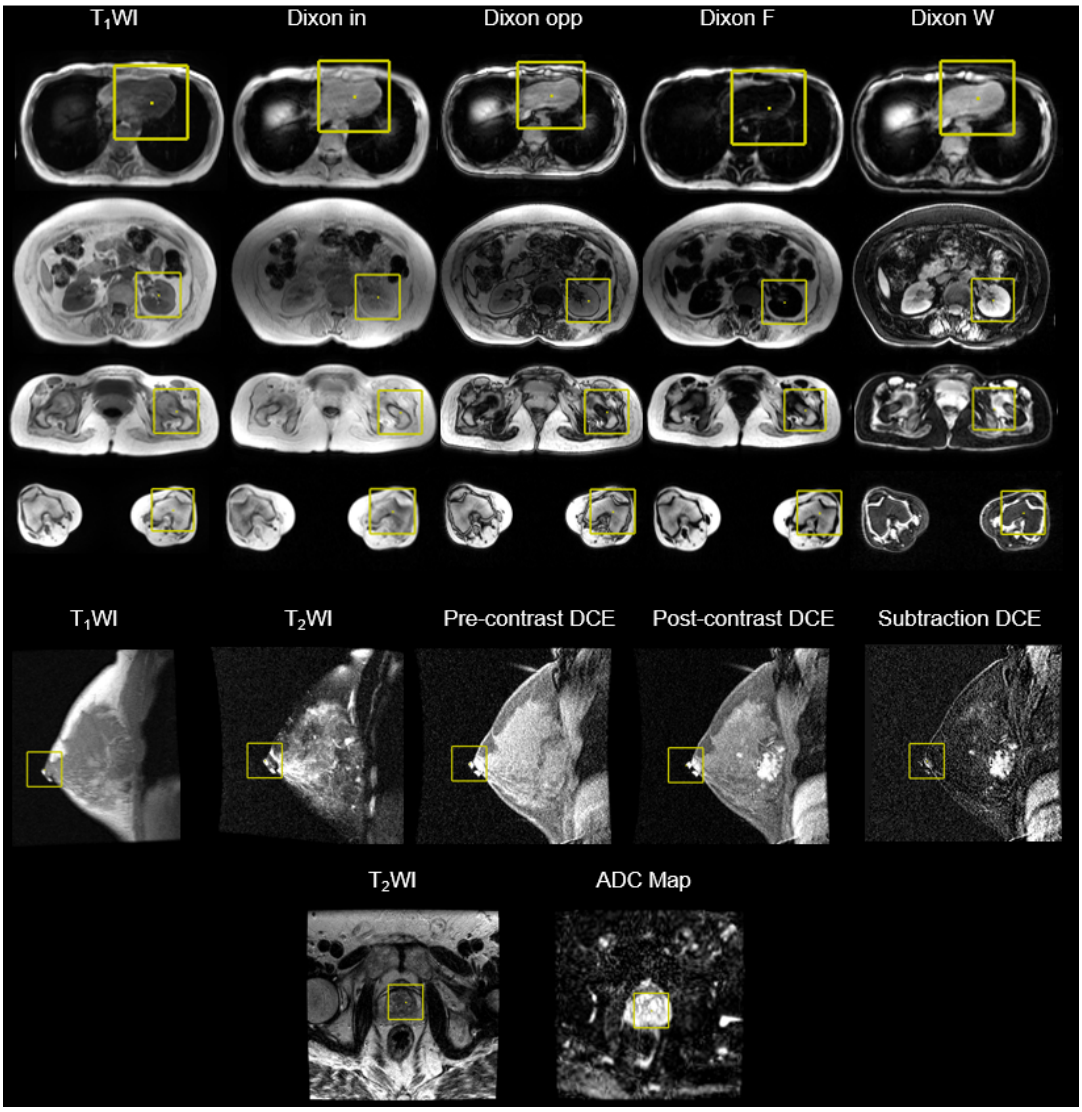

Figure 4.1: Illustration of a deep RL agent trained to localize different anatomical landmarks on a diverse multi-organ dataset with different imaging parameters of T1-weighted (T1WI), T2-weighted(T2WI), Dynamic Contrast Enhanced (DCE), Diffusion Weighted Imaging (DWI) with Apparent Diffusion Coefficient (ADC) mapping and DIXON.

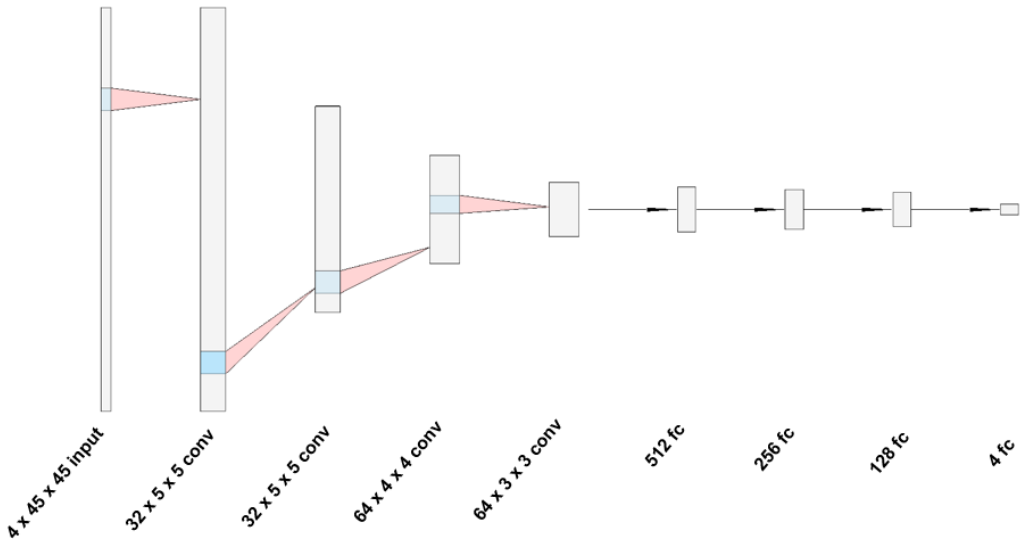

Figure 4.2: The 2D DQN architecture. There is 2x2 maxpooling between each convolutional layer. The last fully connected layer is of size 4 corresponding to the possible actions: move up, down, left, or right.

10. Training lasts for 20 epochs with 25,000 steps per epoch. The target DQN is updated every N = 2,500 steps. The agent follows the multiscale technique with sampling strides 3, 2, 1 corresponding with action step sizes 9, 3, 1.

The complete MIDRL framework was evaluated by dividing the dataset into train and test sets with a 70-30 split. Consequently, the training set consisted of 17 whole body mpMRIs, 9 breast mpMRIs, and 5 prostate mpMRIs, and the test set consisted of 8 whole body mpMRIs, 3 breast mpMRIs, and 3 prostate mpMRIs. The resultant anatomical localization on the test set were evaluated by computing the error in terms of distance between the target and agent's location. The Dice similarity between the terminal agent's and ground truth bounding boxes from each landmark were calculated.

The 3D multi-agent model's network is composed of the same layers as the 2D model but with their 3D counterparts. The convolutional layers of

the network are shared by all agents, but each agent has its own individual set of final fully connected layers. The model has four agents with bounding boxes of size $45 \times 45 \times 45$. Possible actions for an agent include moving in the $\pm x, \pm y, \pm z$ directions (6 actions). An agent's reward is the decrease in Euclidean distance to its target location. Each agent is assigned to locate one landmark (kidney, trochanter, heart, knee). The 3D mutli-agent model was tested on the same whole body imaging parameters (Dixon in phase, Dixon out of phase, Dixon fat, Dixon water, T1WI, T2WI) as in the 2D single agent experiment.

4.5 Evaluation Methods and Results

A total of 57 mpMRIs were evaluated for anatomical localization of breast nipple, prostate, left kidney, left trochanter, left knee, and heart using a diverse set of anatomical locations and mpMRI parameters consisting of T1WI, T2WI, DCE-MRI, DIXON, and DWI. The exact location of the target landmark within an anatomical object was selected as consistently as possible across all images.

The two metrics for reporting model performance are the distance between the agent and target, and the overlap between agent and target bounding boxes.

The overlap between agent and target bounding boxes should be interpreted not only as overlap accuracy for an object, but also as a distance metric. A model's bounding box is fixed in size, but anatomical objects differ in size. Therefore, the amount of extra surrounding area around an object of interest contained within the bounding box may differ among different objects. In

general, the bounding box used in these experiments encompasses the entire object of interest.

4.5.1 Single Agent Evaluation

The single agent model operates in a 2D environment. Therefore, overlap and distance metrics are calculated in 2D for this model. Table 4.1 summarizes the Dice similarity, and Table 4.2 summarizes the distance error for each of the image subsets tested with the 2D single agent model.

4.5.2 Multi-Agent Evaluation

For the multi-agent experiment in the 3D environment, both the agent and target landmark have z-coordinates (slice numbers). It is likely that the agent does not terminate exactly on the target's labeled slice number. Therefore, results for the multi-agent experiment are reported in two methods.

In the first method, performance metrics are calculated in 2D and reported within different ranges of slices centered on the target slice. For example, if a target is located in slice z, then I report results of agents that terminate on a slice within $z \pm 5$. This is a valid method for reporting model performance because the dataset's anatomical landmarks span over multiple slices. I report the agent's and target's bounding boxes' distance and overlap as in the 2D case. Disregarding the z-coordinates, the distance metric is the Euclidean distance between the centers of the agent's and target's bounding boxes in 2D, using only x and y coordinates. Similarly, the overlap metric between bounding boxes is calculated in 2D, using only x and y coordinates. I report the average

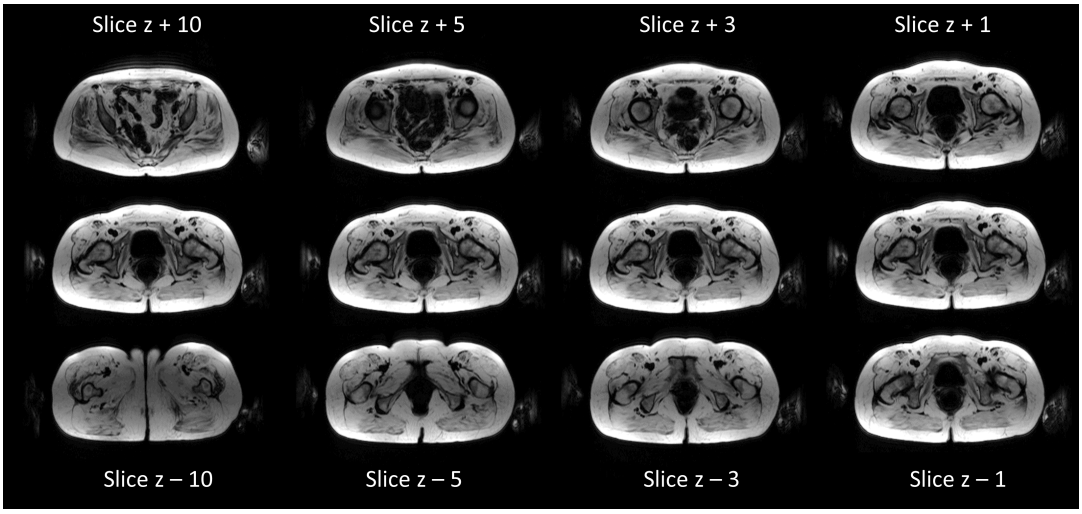

Figure 4.3: An example target slice of the trochanter (middle row) with each group's upper and lower boundary slices for the 3D multi-agent experiment. The target landmark is at slice z. Groups include slices within $z \pm 10$, $z \pm 5$, $z \pm 3$, $z \pm 1$.

and standard deviation of these metrics over four groups of test examples. The four groups are determined as follows: examples where the agent's terminal z-coordinate is 10 or fewer slices from the target's z-coordinate, 5 or fewer, 3 or fewer, and 1 or fewer. These different thresholds represent different levels of accuracy with which the agent located the landmark. Figure 4.3 illustrates these boundary slices. $z \pm 10$ includes slices that do not contain the target object and is a low threshold of accuracy. $z \pm 5$ includes slices that all contain the target object. Therefore, $z \pm 5$ and smaller ranges represent higher thresholds of accuracy. Test examples that do not belong in a group are not included in the group's average and standard deviation calculations. I also report the number of test landmarks that were located by the agent within the z-coordinate threshold out of the total number of evaluated test landmarks. These results are shown in Table 4.3 and Table 4.4. Bounding boxes in this method are of size $45 \times 45 \times 45$.

In the second method, performance metrics are simply reported in 3D without different groupings. Bounding box overlaps are calculated using their entire 3D volume, and distance calculations incorporate the full x,y,z coordinate space. These results are shown in Table 4.5 and Table 4.6. Bounding boxes in this method are of size $45 \times 45 \times 11$ and bound anatomical objects more closely so that 3D overlap metrics are more meaningful.

The 2D distance and overlap metrics demonstrate the multi-agent model's capability in locating general anatomical objects within a 3D whole body volume. The 3D distance and overlap metrics better reflect the multi-agent model's capability in locating a specific landmark (point) within an anatomical object.

Figure 4.4 shows an example of the multi-agent model locating landmarks.

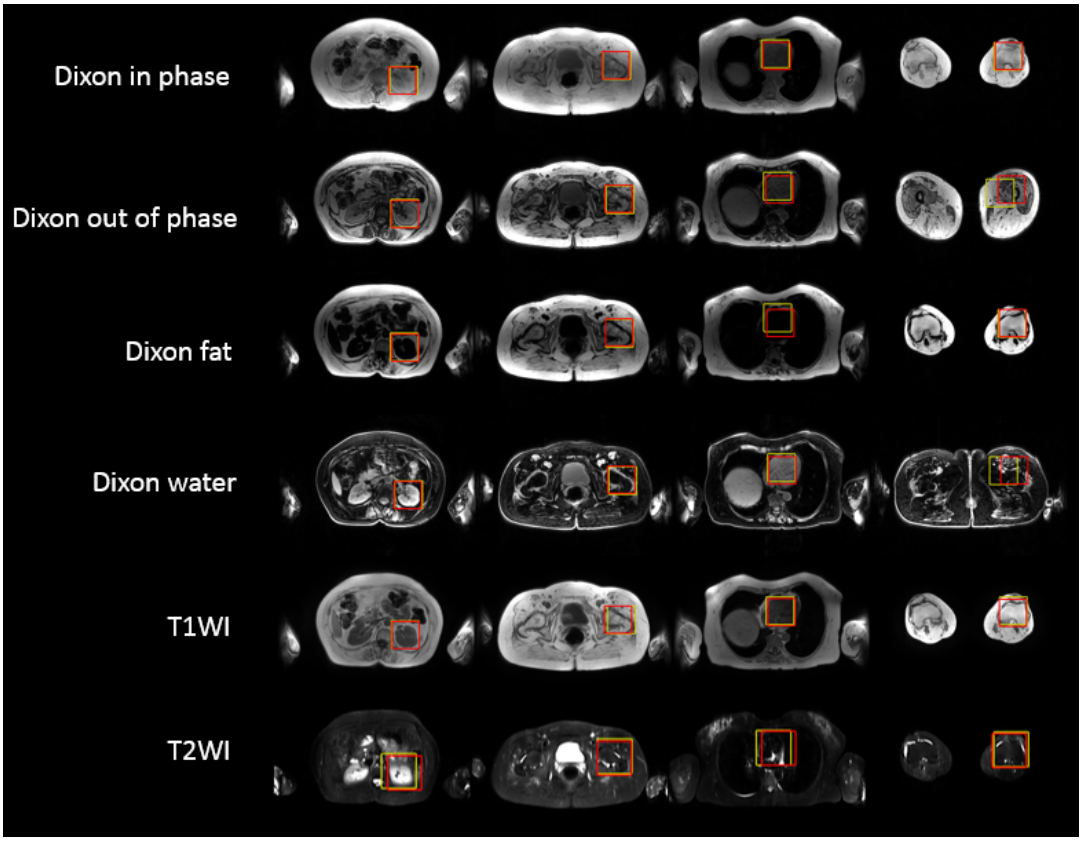

Figure 4.4: An example of the 3D multi-agent model locating landmarks in Dixon in phase, Dixon out of phase, Dixon fat, Dixon water, T1WI, and T2WI whole body images of a patient. Each agent locates one landmark. Landmarks from left to right are the left kidney, left trochanter, heart, and left knee. The agent's bounding box is yellow, and the target's bounding box is red. In this example, the model failed to locate the knee slice in the Dixon out of phase image and the Dixon water image.

Single Agent Dice Scores (2D)

	Heart	**Kidney**	**Trochanter**	**Knee**	**Breast**	**Prostate**
T1WI	0.79 ± 0.19	0.78 ± 0.32	0.83 ± 0.16	0.92 ± 0.05	0.87 ± 0.10	
T2WI	0.74 ± 0.13	0.77 ± 0.10	0.30 ± 0.33	0.86 ± 0.09	0.63 ± 0.27	0.79 ± 0.05
Dixon in	0.88 ± 0.07	0.77 ± 0.21	0.92 ± 0.04	0.93 ± 0.03		
Dixon opp	0.84 ± 0.23	0.82 ± 0.21	0.91 ± 0.05	0.93 ± 0.04		
Dixon F	0.87 ± 0.08	0.80 ± 0.31	0.87 ± 0.08	0.91 ± 0.04		
Dixon W	0.82 ± 0.24	0.89 ± 0.10	0.78 ± 0.20	0.96 ± 0.02		
Post DCE					0.25 ± 0.36	
Pre DCE					0.57 ± 0.41	
Sub DCE					0.17 ± 0.27	
ADC						0.74 ± 0.05
All parameters	0.83 ± 0.14	0.81 ± 0.23	0.79 ± 0.26	0.92 ± 0.06	0.50 ± 0.40	0.77 ± 0.06

Table 4.1: Dice scores (mean $\pm$ stdev) between the 2D single agent's terminal bounding box and the bounding box centered on the target landmark. Both bounding boxes are of size 45×45. A missing entry indicates that the dataset does not include that imaging parameter for that landmark.

Single Agent Distance Errors (mm) in 2D

	Heart	**Kidney**	**Trochanter**	**Knee**	**Breast**	**Prostate**
T1WI	9.8 ± 10.4	23.1 ± 46.2	9.6 ± 10.2	3.5 ± 2.4	3.1 ± 3.4	
T2WI	22.3 ± 17.7	19.0 ± 12.7	68.2 ± 51.8	13.4 ± 5.3	9.5 ± 7.4	20.6 ± 21.2
Dixon in	7.0 ± 4.3	15.1 ± 13.8	5.5 ± 2.2	3.6 ± 1.2		
Dixon opp	10.1 ± 16.8	11.6 ± 12.1	6.4 ± 3.2	3.8 ± 2.7		
Dixon F	7.4 ± 4.8	17.7 ± 30.0	7.9 ± 4.7	3.8 ± 0.7		
Dixon W	6.2 ± 5.6	15.6 ± 24.6	9.8 ± 8.5	2.3 ± 0.7		
Post DCE					30.0 ± 29.5	
Pre DCE					3.3 ± 2.8	
Sub DCE					38.6 ± 28.7	
ADC						16.7 ± 24.8
All parameters	10.0 ± 11.5	16.9 ± 25.5	15.7 ± 27.5	4.5 ± 3.9	16.9 ± 22.9	18.7 ± 20.7

Table 4.2: Distance errors (mean $\pm$ stdev). The distance (in mm) between the 2D single agent's terminal location (center of its bounding box) and the target landmark. A missing entry indicates that the dataset does not include that imaging parameter for that landmark.

Multi-Agent Dice Scores in 2D Overlap

Group	Parameter	Heart	Kidney	Trochanter	Knee
	T1WI	0.89 ± 0.03; 6/8	0.90 ± 0.06; 8/8	0.88 ± 0.10; 8/8	0.89 ± 0.16; 7/8
Agent's	T2WI	0.80 ± 0.13; 2/6	0.82 ± 0.04; 2/6	0.84 ± 0.08; 5/6	0.87 ± 0.04; 5/6
z-coordinate	Dixon in	0.86 ± 0.09; 8/8	0.89 ± 0.06; 8/8	0.92 ± 0.05; 8/8	0.89 ± 0.05; 6/8
within 10 slices	Dixon opp	0.86 ± 0.07; 8/8	0.92 ± 0.04; 8/8	0.88 ± 0.06; 7/8	0.91 ± 0.08; 5/8
from target's	Dixon F	0.79 ± 0.10; 8/8	0.88 ± 0.11; 8/8	0.92 ± 0.03; 8/8	0.95 ± 0.03; 7/8
z-coordinate	Dixon W	0.87 ± 0.04; 8/8	0.89 ± 0.10; 8/8	0.86 ± 0.14; 8/8	0.88 ± 0.08; 6/8
	All Params	0.85 ± 0.08; 40/46	0.89 ± 0.08; 42/46	0.89 ± 0.09; 44/46	0.90 ± 0.09; 36/46
	T1WI	0.89 ± 0.03; 5/8	0.90 ± 0.06; 8/8	0.88 ± 0.10; 8/8	0.89 ± 0.16; 7/8
Agent's	T2WI	0.80 ± 0.13; 2/6	$0.85 \pm$ —; 1/6	0.87 ± 0.06; 4/6	0.86 ± 0.05; 3/6
z-coordinate	Dixon in	0.86 ± 0.09; 8/8	0.89 ± 0.06; 8/8	0.92 ± 0.05; 8/8	0.89 ± 0.05; 6/8
within 5 slices	Dixon opp	0.86 ± 0.07; 8/8	0.92 ± 0.04; 8/8	0.88 ± 0.06; 7/8	0.95 ± 0.02; 4/8
from target's	Dixon F	0.80 ± 0.10; 6/8	0.88 ± 0.11; 8/8	0.92 ± 0.03; 8/8	0.95 ± 0.03; 7/8
z-coordinate	Dixon W	0.88 ± 0.04; 7/8	0.89 ± 0.10; 8/8	0.86 ± 0.14; 8/8	0.91 ± 0.04; 5/8
	All Params	0.85 ± 0.07; 36/46	0.90 ± 0.08; 41/46	0.89 ± 0.08; 43/46	0.91 ± 0.08; 32/46
	T1WI	0.88 ± 0.03; 4/8	0.91 ± 0.06; 6/8	0.88 ± 0.10; 8/8	0.95 ± 0.02; 5/8
Agent's	T2WI	$0.89 \pm$ —; 1/6	$0.85 \pm$ —; 1/6	0.87 ± 0.06; 4/6	0.86 ± 0.05; 3/6
z-coordinate	Dixon in	0.90 ± 0.03; 5/8	0.89 ± 0.06; 8/8	0.92 ± 0.05; 8/8	0.90 ± 0.04; 5/8
within 3 slices	Dixon opp	0.84 ± 0.08; 5/8	0.92 ± 0.04; 8/8	0.88 ± 0.06; 7/8	0.95 ± 0.02; 4/8
from target's	Dixon F	0.80 ± 0.10; 6/8	0.87 ± 0.12; 7/8	0.92 ± 0.03; 8/8	0.95 ± 0.03; 7/8
z-coordinate	Dixon W	0.88 ± 0.04; 7/8	0.89 ± 0.11; 7/8	0.87 ± 0.15; 7/8	0.93 ± 0.02; 4/8
	All Params	0.86 ± 0.07; 28/46	0.89 ± 0.08; 37/46	0.89 ± 0.08; 42/46	0.93 ± 0.04; 28/46
	T1WI	0.90 ± 0.01; 2/8	0.89 ± 0.06; 4/8	0.91 ± 0.03; 6/8	0.96 ± 0.02; 3/8
Agent's	T2WI	— $\pm$ —; 0/8	— $\pm$ —; 0/8	0.90 ± 0.05; 3/6	$0.81 \pm$ —; 1/6
z-coordinate	Dixon in	0.89 ± 0.03; 4/8	0.90 ± 0.04; 5/8	0.92 ± 0.05; 8/8	0.93 ± 0.02; 3/8
within 1 slice	Dixon opp	0.88 ± 0.04; 3/8	0.95 ± 0.02; 5/8	0.88 ± 0.07; 5/8	0.95 ± 0.02; 4/8
from target's	Dixon F	0.82 ± 0.10; 2/8	0.93 ± 0.05; 3/8	0.92 ± 0.03; 8/8	0.96 ± 0.02; 5/8
z-coordinate	Dixon W	0.86 ± 0.02; 4/8	0.93 ± 0.06; 4/8	0.92 ± 0.03; 5/8	0.93 ± 0.02; 3/8
	All Params	0.87 ± 0.05; 15/46	0.92 ± 0.05; 21/46	0.91 ± 0.05; 35/46	0.94 ± 0.04; 19/46

Table 4.3: 2D Dice scores for the 3D whole body multi-agent experiment (mean $\pm$ stdev; number of examples where the agent's z-coordinate was close to the target's z-coordinate / total number of examples). Bounding box sizes for calculating overlap are 45×45. Test samples that are not located within the group's range by the agent are not included in the mean and stdev calculations for the corresponding entry.

Multi-Agent Distance Errors (mm) in 2D

Group	Parameter	Heart	Kidney	Trochanter	Knee
Agent's z-coordinate within 10 slices from target's z-coordinate	T1WI	6.6 ± 2.1; 6/8	5.8 ± 4.3; 8/8	7.8 ± 6.4; 8/8	6.9 ± 10.9; 7/8
	T2WI	10.5 ± 6.9; 2/6	9.7 ± 2.4; 2/6	9.1 ± 5.5; 5/6	6.9 ± 1.7; 5/6
	Dixon in	7.8 ± 4.7; 8/8	6.8 ± 3.6; 8/8	5.3 ± 2.9; 8/8	6.2 ± 2.9; 6/8
	Dixon opp	7.5 ± 3.6; 8/8	4.2 ± 2.8; 8/8	7.5 ± 2.8; 7/8	4.8 ± 4.5; 5/8
	Dixon F	11.7 ± 6.0; 8/8	6.8 ± 6.3; 8/8	4.9 ± 2.0; 8/8	3.3 ± 2.2; 7/8
	Dixon W	7.6 ± 3.1; 8/8	6.5 ± 5.5; 8/8	8.4 ± 9.2; 8/8	6.7 ± 4.8; 6/8
	All Params	8.4 ± 4.4; 40/46	6.2 ± 4.5; 42/46	7.0 ± 5.4; 44/46	5.8 ± 5.5; 36/46
Agent's z-coordinate within 5 slices from target's z-coordinate	T1WI	6.7 ± 2.4; 5/8	5.8 ± 4.3; 8/8	7.8 ± 6.4; 8/8	6.9 ± 10.9; 7/8
	T2WI	10.5 ± 6.9; 2/6	8.0 ± —; 1/6	7.2 ± 4.3; 4/6	7.5 ± 2.0; 3/6
	Dixon in	7.8 ± 4.7; 8/8	6.8 ± 3.6; 8/8	5.3 ± 2.9; 8/8	6.2 ± 2.9; 6/8
	Dixon opp	7.5 ± 3.6; 8/8	4.2 ± 2.8; 8/8	7.5 ± 2.8; 7/8	2.9 ± 1.4; 4/8
	Dixon F	11.0 ± 5.4; 6/8	6.8 ± 6.3; 8/8	4.9 ± 2.0; 8/8	3.3 ± 2.2; 7/8
	Dixon W	7.3 ± 3.1; 7/8	6.5 ± 5.5; 8/8	8.4 ± 9.2; 8/8	5.0 ± 3.0; 5/8
	All Params	8.2 ± 4.1; 36/46	6.1 ± 4.5; 41/46	6.8 ± 5.2; 43/46	5.3 ± 5.5; 32/46
Agent's z-coordinate within 3 slices from target's z-coordinate	T1WI	6.8 ± 2.7; 4/8	4.6 ± 3.8; 6/8	7.8 ± 6.4; 8/8	2.6 ± 1.0; 5/8
	T2WI	5.6 ± —; 1/6	8.0 ± —; 1/6	7.2 ± 4.3; 4/6	7.5 ± 2.0; 3/6
	Dixon in	5.9 ± 1.7; 5/8	6.8 ± 3.6; 8/8	5.3 ± 2.9; 8/8	5.5 ± 2.6; 5/8
	Dixon opp	8.4 ± 4.4; 5/8	4.2 ± 2.8; 8/8	7.5 ± 2.8; 7/8	2.9 ± 1.4; 4/8
	Dixon F	11.0 ± 5.4; 6/8	7.3 ± 6.6; 7/8	4.9 ± 2.0; 8/8	3.3 ± 2.2; 7/8
	Dixon W	7.3 ± 3.1; 7/8	6.5 ± 6.0; 7/8	7.9 ± 9.8; 7/8	3.9 ± 2.0; 4/8
	All Params	7.9 ± 3.8; 28/46	5.9 ± 4.6; 37/46	6.7 ± 5.2; 42/46	4.1 ± 2.4; 28/46
Agent's z-coordinate within 1 slice from target's z-coordinate	T1WI	4.6 ± 1.5; 2/8	5.8 ± 3.9; 4/8	6.1 ± 2.4; 6/8	2.7 ± 1.0; 3/8
	T2WI	— ± —; 0/8	— ± —; 0/8	6.0 ± 4.3; 3/6	9.9 ± —; 1/6
	Dixon in	6.1 ± 1.9; 4/8	6.0 ± 2.6; 5/8	5.3 ± 2.9; 8/8	4.0 ± 2.1; 3/8
	Dixon opp	6.5 ± 3.7; 3/8	2.9 ± 2.0; 5/8	7.4 ± 3.4; 5/8	2.9 ± 1.4; 4/8
	Dixon F	10.0 ± 4.8; 2/8	4.0 ± 2.2; 3/8	4.9 ± 2.0; 8/8	2.4 ± 0.9; 5/8
	Dixon W	8.6 ± 2.9; 4/8	4.1 ± 2.5; 4/8	4.7 ± 1.9; 5/8	3.8 ± 2.4; 3/8
	All Params	7.2 ± 3.1; 15/46	4.6 ± 2.7; 21/46	5.6 ± 2.6; 35/46	3.4 ± 2.1; 19/46

Table 4.4: 2D distance error (x,y distance) in mm for the 3D whole body multi-agent experiment (mean ± stdev; number of test examples where the agent's z-coordinate was close to the target's z-coordinate / total number of test examples). Test samples that are not located within the group's range by the agent are not included in the mean and stdev calculations for the corresponding entry.

Multi-Agent Dice Scores in 3D

	Heart	Kidney	Trochanter	Knee
T1WI	0.74 ± 0.18	0.78 ± 0.14	0.87 ± 0.07	0.53 ± 0.35
T2WI	0.63 ± 0.13	0.54 ± 0.36	0.46 ± 0.29	0.43 ± 0.40
Dixon in	0.83 ± 0.10	0.66 ± 0.28	0.89 ± 0.07	0.52 ± 0.44
Dixon opp	0.75 ± 0.14	0.82 ± 0.10	0.85 ± 0.08	0.56 ± 0.46
Dixon F	0.60 ± 0.30	0.62 ± 0.32	0.91 ± 0.61	0.77 ± 0.31
Dixon W	0.78 ± 0.10	0.85 ± 0.10	0.71 ± 0.25	0.54 ± 0.45
All parameters	0.73 ± 0.18	0.72 ± 0.25	0.80 ± 0.21	0.56 ± 0.40

Table 4.5: 3D Dice scores (mean ± stdev) between the multi-agent's terminal bounding box and the bounding box centered on the target landmark. Bounding boxes are of size $45 \times 45 \times 11$ and generally encompass the entire object of interest.

Multi-Agent Distance Errors (mm) in 3D

	Heart	Kidney	Trochanter	Knee
T1WI	12.6 ± 9.4	10.2 ± 6.5	6.4 ± 2.7	22.9 ± 20.4
T2WI	18.0 ± 7.6	29.3 ± 33.2	27.6 ± 17.1	91.7 ± 126.2
Dixon in	8.1 ± 4.7	18.3 ± 21.8	5.5 ± 2.6	51.6 ± 64.0
Dixon opp	11.4 ± 6.5	8.5 ± 4.5	7.0 ± 3.2	57.4 ± 78.5
Dixon F	21.7 ± 17.9	39.2 ± 72.0	4.2 ± 2.4	32.3 ± 75.9
Dixon W	10.1 ± 4.8	6.9 ± 4.0	15.9 ± 14.5	68.7 ± 92.6
All parameters	13.5 ± 10.3	18.3 ± 34.1	10.4 ± 11.4	52.5 ± 78.4

Table 4.6: 3D distance error (x,y,z distance) in mm for the 3D whole body multi-agent experiment (mean $\pm$ stdev).

4.6 Discussion

The 2D single agent model performed well overall in locating the heart, kidney, trochanter, knee, and prostate, but not as well in locating the nipple in the breast images. In the breast images, the spiculations in the tumor were often incorrectly identified due to their shape being similar to the area around the nipple. The model also performed poorly on the T2WI trochanter possibly due to the image's proximity and similarity to the prostate.

The results for the 2D metrics described in Section 4.5.2 show that the multi-agent model was able to locate anatomical objects in the 3D whole body environment across all imaging parameters. Among the agents that terminated close to the target slice (similar z-coordinates), overlap and distance metrics were similar to those in the 2D single agent experiment. The model performed particularly well on Dixon images, on which most landmarks were located within five slices of their labeled slice number. The model did not perform as well on T2WI as compared to the rest of the imaging parameters. Anatomical objects are present in the slices spanned by $z \pm 5$, and the model was able to locate most objects within this range. While results for slices spanned by $z \pm 10$ are reported, these results are not as representative

of the model's true performance because not all slices in this range contain the anatomical object.

The results for the 3D metrics described in Section 4.5.2 demonstrate that the multi-agent model was able to locate specific points within the 3D whole body environment across all imaging parameters. The model did not perform as well in locating the exact target location in the knee. Oftentimes, the model would locate the bone in a thigh slice instead of the knee.

The MIDRL results demonstrate the feasibility of training deep RL agents for anatomical localization across a diverse set of anatomical environments and imaging parameters with excellent results. These results assert the possibility of a general AI framework based on deep RL for anatomical landmark localization in radiological applications.

Previously, multi-agent deep reinforcement learning with collaborating agents were evaluated in the radiological setting for detection of multiple landmarks in brain MRI, cardiac MRI, and fetal brain ultrasound with excellent results [58]. However, these models were limited to a single anatomical environment and not in a multi-environment framework. In contrast, I successfully trained deep RL agents to localize different anatomical landmarks across diverse datasets that were acquired with different fields of view, magnetic field strengths (1.5T and 3T), imaging orientations (sagittal and axial), and imaging parameters (T1WI, T2WI, DCE, DWI, DIXON), demonstrating the robustness of the MIDRL framework.

Training a deep reinforcement learning model is a computationally expensive process, making individual landmark detection impractical for clinical

translation. This work demonstrates the multitask capability of deep RL agents in a diverse environment with excellent success, thereby improving upon both the space and time complexity of existing deep RL frameworks.

In conclusion, MIDRL initiates a first step towards a universal deep reinforcement learning framework capable of locating anatomical landmarks irrespective of the imaging modality, underlying anatomical environment, and image acquisition parameters. Automatic localization of landmarks could be used in several medical image analysis applications, such as in algorithms for image registration and for image segmentation.

Chapter 5

Conclusion

This thesis presents multiparametric deep learning (MPDL) approaches to medical image segmentation. Patch-based CNNs and stacked sparse autoencoders use the contrast provided by multiple imaging parameters to segment 3D MRIs of patients with limb girdle muscular dystrophy into five classes: background, muscle, bone, fat, and fat infiltration. There exists no prior work in segmentation of MRIs of patients with muscular dystrophy. Excellent segmentation accuracy is achieved in slices of the thighs. The fact that the models are able to learn from a relatively small dataset of manually labeled tissue signatures is noteworthy. Even when fully labeled datasets are not available, which is often the case in a hospital setting, the methods described in this work are a viable approach. The models developed in this thesis have been used to obtain quantitative metrics for tracking the progression of muscular dystrophy in patients as part of a clinical trial performed at Johns Hopkins School of Medicine.

The reinforcement learning (RL) methods presented in this work serve as an initial step toward improving segmentation performance. The RL models

efficiently and accurately locate multiple anatomical landmarks across multiple imaging parameters in 2D and 3D MRIs of both patients and healthy subjects. In future work, spatial features obtained by the RL models, such as location within the entire image, could be incorporated into segmentation models such as the MPDL models. This would provide MDPL models with broader image context to better discern different objects of similar contrast. Automatic landmark localization could also be used to locate landmarks for automatic image registration.

Bibliography

[1] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. "Snakes: Active contour models". In: *International journal of computer vision* 1.4 (1988), pp. 321–331.

[2] Laurent D Cohen. "On active contour models and balloons". In: *CVGIP: Image understanding* 53.2 (1991), pp. 211–218.

[3] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. "Geodesic active contours". In: *International journal of computer vision* 22.1 (1997), pp. 61–79.

[4] Chenyang Xu and Jerry L Prince. "Snakes, shapes, and gradient vector flow". In: *IEEE Transactions on image processing* 7.3 (1998), pp. 359–369.

[5] James C Gee, Martin Reivich, and Ruzena Bajcsy. "Elastically deforming a three-dimensional atlas to match anatomical brain images". In: (1993).

[6] D Louis Collins, Colin J Holmes, Terrence M Peters, and Alan C Evans. "Automatic 3-D model-based neuroanatomical segmentation". In: *Human brain mapping* 3.3 (1995), pp. 190–208.

[7] Torsten Rohlfing, Robert Brandt, Randolf Menzel, and Calvin R Maurer Jr. "Evaluation of atlas selection strategies for atlas-based image segmentation with application to confocal microscopy images of bee brains". In: *NeuroImage* 21.4 (2004), pp. 1428–1442.

[8] Yuri Boykov and Marie-Pierre Jolly. "Interactive organ segmentation using graph cuts". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2000, pp. 276–286.

[9] Yuri Y Boykov and M-P Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images". In: *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 1. IEEE. 2001, pp. 105–112.

[10] Xiang Lin, Brett Cowan, and Alistair Young. "Model-based graph cut method for segmentation of the left ventricle". In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. IEEE. 2006, pp. 3059–3062.

[11] Xinjian Chen, Jayaram K Udupa, Ulas Bagci, Ying Zhuge, and Jianhua Yao. "Medical image segmentation by combining graph cuts and oriented active appearance models". In: *IEEE Transactions on Image Processing* 21.4 (2012), pp. 2035–2046.

[12] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[13] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[16] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[18] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[19] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[20] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. "Deep neural networks segment neuronal membranes in electron microscopy images". In: *Advances in neural information processing systems*. 2012, pp. 2843–2851.

[21] Feng Ning, Damien Delhomme, Yann LeCun, Fabio Piano, Léon Bottou, and Paolo Emilio Barbano. "Toward automatic phenotyping of developing embryos from videos". In: *IEEE Transactions on Image Processing* 14.9 (2005), pp. 1360–1371.

[22] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.

[23] Maximilian Riesenhuber and Tomaso Poggio. "Hierarchical models of object recognition in cortex". In: *Nature neuroscience* 2.11 (1999), pp. 1019–1025.

[24] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. "Unsupervised learning of invariant feature hierarchies with applications to object recognition". In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE. 2007, pp. 1–8.

[25] Le Hou, Dimitris Samaras, Tahsin M Kurc, Yi Gao, James E Davis, and Joel H Saltz. "Patch-based convolutional neural network for whole slide tissue image classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2424–2433.

[26] Wen Li et al. "Automatic segmentation of liver tumor in CT images with deep convolutional neural networks". In: *Journal of Computer and Communications* 3.11 (2015), p. 146.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[28] Avi Ben-Cohen, Idit Diamant, Eyal Klang, Michal Amitai, and Hayit Greenspan. "Fully convolutional network for liver segmentation and lesions detection". In: *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 77–85.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[30] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[31] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 565–571.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[33] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[34] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 11–19.

[35] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[36] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[37] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1520–1528.

[38] Jun Fu, Jing Liu, Yuhang Wang, Jin Zhou, Changyong Wang, and Hanqing Lu. "Stacked deconvolutional network for semantic segmentation". In: *IEEE Transactions on Image Processing* (2019).

[39] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[40] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[41] Jun Xu, Lei Xiang, Qingshan Liu, Hannah Gilmore, Jianzhong Wu, Jinghai Tang, and Anant Madabhushi. "Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images". In: *IEEE transactions on medical imaging* 35.1 (2015), pp. 119–130.

[42] GB Praveen, Anita Agrawal, Ponraj Sundaram, and Sanjay Sardesai. "Ischemic stroke lesion segmentation using stacked sparse autoencoder". In: *Computers in biology and medicine* 99 (2018), pp. 38–52.

[43] Vishwa S Parekh, Katarzyna J Macura, Susan C Harvey, Ihab R Kamel, Riham EI-Khouli, David A Bluemke, and Michael A Jacobs. "Multiparametric deep learning tissue signatures for a radiological biomarker of breast cancer: Preliminary results". In: *Medical physics* 47.1 (2020), pp. 75–88.

[44] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2nd ed. MIT press, 2018.

[45] Farhang Sahba, Hamid R Tizhoosh, and Magdy MA Salama. "A reinforcement learning framework for medical image segmentation". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE. 2006, pp. 511–517.

[46] Farhang Sahba, Hamid R Tizhoosh, and Magdy MMA Salama. "Application of opposition-based reinforcement learning in image segmentation". In: *2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing*. IEEE. 2007, pp. 246–251.

[47] Hamid R Tizhoosh. "Opposition-based reinforcement learning". In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* 10.3 (2006).

[48] Farhang Sahba, Hamid R Tizhoosh, and Magdy MA Salama. "Application of reinforcement learning for segmentation of transrectal ultrasound images". In: *BMC medical imaging* 8.1 (2008), p. 8.

[49] Florin C Ghesu, Bogdan Georgescu, Tommaso Mansi, Dominik Neumann, Joachim Hornegger, and Dorin Comaniciu. "An artificial agent for anatomical landmark detection in medical images". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 229–237.

[50] Florin-Cristian Ghesu, Bogdan Georgescu, Yefeng Zheng, Sasa Grbic, Andreas Maier, Joachim Hornegger, and Dorin Comaniciu. "Multi-scale deep reinforcement learning for real-time 3D-landmark detection in CT scans". In: *IEEE transactions on pattern analysis and machine intelligence* 41.1 (2017), pp. 176–189.

[51] Florin C Ghesu, Bogdan Georgescu, Sasa Grbic, Andreas Maier, Joachim Hornegger, and Dorin Comaniciu. "Towards intelligent robust detection of anatomical structures in incomplete volumetric data". In: *Medical image analysis* 48 (2018), pp. 203–213.

[52] Amir Alansary, Ozan Oktay, Yuanwei Li, Loic Le Folgoc, Benjamin Hou, Ghislain Vaillant, Konstantinos Kamnitsas, Athanasios Vlontzos, Ben Glocker, Bernhard Kainz, et al. "Evaluating reinforcement learning agents for anatomical landmark detection". In: *Medical image analysis* 53 (2019), pp. 156–164.

[53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[54] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[55] Hado V Hasselt. "Double Q-learning". In: *Advances in neural information processing systems*. 2010, pp. 2613–2621.

[56] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning". In: *Thirtieth AAAI conference on artificial intelligence*. 2016.

[57] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. "Dueling network architectures for deep reinforcement learning". In: *arXiv preprint arXiv:1511.06581* (2015).

[58] Athanasios Vlontzos, Amir Alansary, Konstantinos Kamnitsas, Daniel Rueckert, and Bernhard Kainz. "Multiple Landmark Detection using Multi-Agent Reinforcement Learning". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 262–270.

[59] Rui Liao, Shun Miao, Pierre de Tournemire, Sasa Grbic, Ali Kamen, Tommaso Mansi, and Dorin Comaniciu. "An artificial agent for robust image registration". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[60] Kai Ma, Jiangping Wang, Vivek Singh, Birgi Tamersoy, Yao-Jen Chang, Andreas Wimmer, and Terrence Chen. "Multimodal image registration with deep context reinforcement learning". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 240–248.

[61] Julian Krebs, Tommaso Mansi, Hervé Delingette, Li Zhang, Florin C Ghesu, Shun Miao, Andreas K Maier, Nicholas Ayache, Rui Liao, and Ali Kamen. "Robust non-rigid registration through agent-based action learning". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 344–352.

[62] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[63] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. "A survey on deep learning in medical image analysis". In: *Medical image analysis* 42 (2017), pp. 60–88.

[64] Vishwa S Parekh, John Laterra, Chetan Bettegowda, Alex E Bocchieri, Jay J Pillai, and Michael A Jacobs. "Multiparametric Deep Learning and Radiomics for Tumor Grading and Treatment Response Assessment of Brain Cancer: Preliminary Results". In: *arXiv preprint arXiv:1906.04049* (2019).

[65] Vishwa Parekh and Michael A Jacobs. "Radiomics: a new application from established techniques". In: *Expert review of precision medicine and drug development* 1.2 (2016), pp. 207–226.

[66] Vishwa S Parekh and Michael A Jacobs. "Integrated radiomic framework for breast cancer and tumor biology using advanced machine learning and multiparametric MRI". In: *NPJ breast cancer* 3.1 (2017), pp. 1–9.

[67] Matthew P Wicklund and John T Kissel. "The limb-girdle muscular dystrophies". In: *Neurologic clinics* 32.3 (2014), pp. 729–749.

[68] Joe P Windham, Mahmoud A Abd-Allah, David A Reimann, Jerry W Froelich, and Allan M Haggar. "Eigenimage filtering in MR imaging." In: *Journal of computer assisted tomography* 12.1 (1988), pp. 1–9.

[69] Doris G Leung, John A Carrino, Kathryn R Wagner, and Michael A Jacobs. "Whole-body magnetic resonance imaging evaluation of facioscapulohumeral muscular dystrophy". In: *Muscle & nerve* 52.4 (2015), pp. 512–520.

[70] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. "Deep reinforcement learning for dialogue generation". In: *arXiv preprint arXiv:1606.01541* (2016).

[71] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), pp. 354–359.

[72] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. "Deep reinforcement learning framework for autonomous driving". In: *Electronic Imaging* 2017.19 (2017), pp. 70–76.

Vita

Alex Bocchieri is a master's student in Computer Science at Johns Hopkins University (JHU). He received his BS in Electrical Engineering in 2018 and his MSE in Electrical and Computer Engineering in 2019, both from JHU. He is broadly interested in machine learning and signal processing.

Alex has been doing research in machine learning and medical image analysis under the supervision of Dr. Michael Jacobs and Dr. Vladimir Braverman, which resulted in this thesis. During this time, he published one journal paper, one conference paper, one conference abstract, and one paper on arXiv.

As an undergraduate, Alex worked under Dr. Brinton Cooper on an error control coding project. Alex worked at JHU Applied Physics Laboratory over the summers of 2017 and 2018 as a technical intern.